Journal of Intelligent Information Systems, 14, 175–198, 2000
© 2000 Kluwer Academic Publishers. Printed in The Netherlands.

Machine Learning for Intelligent Processing of Printed Documents

FLORIANA ESPOSITO esposito@di.uniba.it
DONATO MALERBA malerba@di.uniba.it
FRANCESCA A. LISI lisi@di.uniba.it
Dipartimento di Informatica, Università degli Studi di Bari, via Orabona 4, 70125 Bari, Italy

Abstract. A paper document processing system is an information system component which transforms information on printed or handwritten documents into a computer-revisable form. In intelligent systems for paper document processing this information capture process is based on knowledge of the specific layout and logical structures of the documents. This article proposes the application of machine learning techniques to acquire the specific knowledge required by an intelligent document processing system, named WISDOM++, that manages printed documents, such as letters and journals. Knowledge is represented by means of decision trees and first-order rules automatically generated from a set of training documents. In particular, an incremental decision tree learning system is applied for the acquisition of decision trees used for the classification of segmented blocks, while a first-order learning system is applied for the induction of rules used for the layout-based classification and understanding of documents. Issues concerning the incremental induction of decision trees and the handling of both numeric and symbolic data in first-order rule learning are discussed, and the validity of the proposed solutions is empirically evaluated by processing a set of real printed documents.

Keywords: learning and knowledge discovery, intelligent information systems, intelligent document processing, decision-tree learning, first-order rule induction

1. Introduction

One of the key issues regarding the use of information systems is the acquisition of new information, which often resides in paper documents. In order to provide a suitable solution to this problem, information systems will have to be integrated with paper document processing systems, which are devised to transform printed or handwritten documents into a computer-revisable form. Since the 1960's, much research on paper document processing has focused on optical character recognition (OCR). In the last decade, it has been widely recognized that text acquisition by means of OCR is only one step of document processing, which also includes the separation of text from graphics, the classification of documents, the identification (or semantic labelling) of some relevant components of the page layout and the transformation of the document into an electronic format. In the literature, the process of breaking down the bitmap of a scanned paper document (*document image*) into several layout components is called *document analysis*, while the process of attaching semantic (or logic) labels to some layout components is named *document understanding* (Tang et al., 1994). Furthermore, the term *document classification* has been introduced to identify the process of attaching a semantic label (a class name) to the whole document (see figure 1) (Esposito et al., 1990).

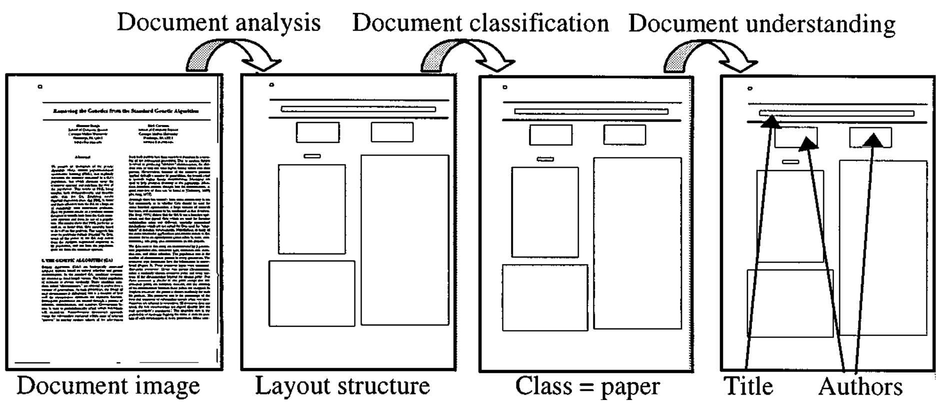


Figure 1. Three document processing steps: a) document analysis, that is breaking down the bitmap of a document image into several layout components; b) document classification, that is assigning the document to a pre-defined set of classes; c) document understanding, that is attaching semantic (or logic) labels to some layout components.

Intelligent document processing systems require specific *knowledge* for the analysis, classification and understanding of paper documents. For instance, the image segmentation can be based on the layout conventions (or *layout structure*) of specific classes of documents, while the separation of text from graphics requires knowledge on how text blocks can be distinguished from non-text blocks. The importance of knowledge management in document processing has led some authors to define document analysis and understanding as a branch of artificial intelligence (Tang et al., 1994). In many applications presented in the literature, a great effort is made to hand-code the necessary knowledge according to some formalism (e.g., block grammars (Nagy et al., 1992), geometric trees (Dengel and Barth, 1989), and frames (Bayer et al., 1994)). Such hand-coding is time-consuming and limits the application of intelligent document processing systems to pre-defined classes of documents. In this article we advocate an extensive application of machine learning techniques and tools in order to solve the knowledge acquisition bottleneck problem. This approach has been pursued in the design and development of an intelligent document processing system, named *WISDOM++*, which is a newer object-oriented version of the system WISDOM (**W**indows **I**nterface **S**ystem for **DO**cument **M**anagement) (Malerba et al., 1997b), originally written in C and used to feed a digital library (Esposito et al., 1998). The two main requirements considered in the design of WISDOM++ are real-time user interaction and adaptivity. The former involves choosing fast algorithms for document image analysis, while the latter requires the application of machine learning techniques.

This application to document processing has some peculiarities (see figure 2). Firstly, it is important to clearly formulate the problems as well as the learning tasks (e.g., classification, clustering, regression, etc.) for the particular application domain. In our work, three problems have been identified in the whole document process:

1. Classification of *blocks* defined by the segmentation algorithm, in order to separate text from non-text areas in the document image.

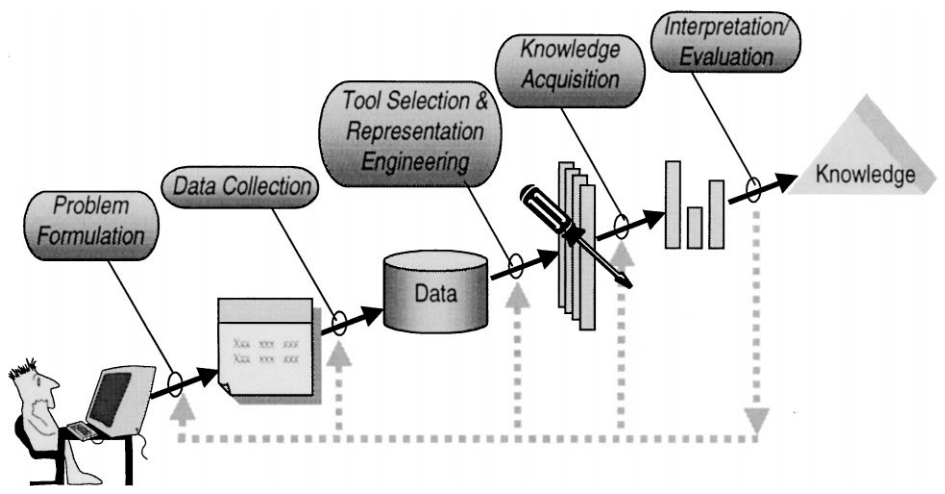

Problem Formulation
Data Collection
Tool Selection & Representation Engineering
Knowledge Acquisition
Interpretation/ Evaluation
Data
Knowledge

Figure 2. Stages of the application of machine learning techniques to paper document processing (adapted from Fayyad et al. (1996)).

2. Assignment of documents to one of a pre-defined set of classes (document classification).
3. Association of semantic (or logic) labels to some layout components (document understanding).

These are problems in which machine learning techniques can be effectively applied, and correspond to some of the document processing steps (see next section). For all of them, the main task is classification.

The second aspect of the application is the creation of the target data set. In this article, we consider a set of real, single-page documents, which correspond to the photocopy of first-pages of articles published either in conference proceedings or in professional journals. This choice is due to our interest in feeding a prototype of an intelligent digital library (Esposito et al., 1998), although previous works have shown that machine learning techniques can be applied to a wide range of paper documents, including letters (Esposito et al., 1994) and maps (Esposito et al., 1997). It is noteworthy that in the case of document classification a training example is a single-page document, whereas in the block classification a training example is a block defined by the segmentation algorithm, and in document understanding a training example is a layout component (*frame2*) defined by the layout analysis process (see next section). Thus, a single document is a source of one or more training examples, according to the particular learning problem.

The third aspect of the application is the choice of the best representation of training examples. For the first of the three learning problems, experimental results reported in Section 3 show that a feature vector representation of blocks seems to be the most appropriate. Indeed, the pattern of pixels in a block is not deemed relevant in other papers (Wong et al., 1982; Fisher et al., 1990), and in some cases only the texture of the block is considered (Wang and Srihari, 1989). Regarding document classification, training example representation depends on the information already extracted in previous processing steps. For instance, in the work by Pagurek et al. (1990), the textual content of a document is important to correctly classify the document itself. This implies the use of time-consuming OCR techniques to "read" the content. On the contrary, in our work the only information considered for document classification is the layout structure. Our approach is based on the idea that humans are generally able to classify documents (invoices, letters, order forms, papers, indexes, etc.) from a perceptive point of view, by recognizing the layout structure of a form. This means that documents belonging to the same class have a set of relevant and invariant layout characteristics, called *page layout signature*, which can be used for classification. However, the representation of such a page layout signature requires first-order logic formalisms or equivalents (e.g., attributed graphs), due to the presence of geometric relations among layout components. The same applies to the document understanding problem, since we assume that the identification of layout components with a logical meaning can be based on their geometric properties and relations. Other representation issues regarding the document processing domain are the accurate preprocessing of document images in order to reduce the effect of noise and the consideration of both numeric and symbolic descriptions, since the former increase the sensitivity while the latter increase the stability of the internal representation of images (Connell and Brady, 1987).

The fourth aspect of the application is the choice of appropriate machine learning tools and techniques. In previous works on block classification, linear discriminant analysis techniques were used (Wang and Srihari, 1989; Wong et al., 1982). It is well-known that these parametric statistical techniques cannot handle complicated non-linear interactions among features. On the contrary, the top-down induction of decision trees does handle such interactions and produces results that are simple to interpret. In fact, this is the approach followed in our work (see Section 3). To solve document classification and understanding problems, it is necessary to resort to first-order learning systems, suitably extended in order to handle both numeric and symbolic attributes and relations (see Section 4).

The fifth and last stage of the application of machine learning techniques to document processing is the evaluation and interpretation of results. Predictive accuracy is certainly the most important evaluation criterion for the three learning problems considered in this work, but it is not the only one. When possible, it is important to distinguish *omission* errors from *commission* errors. Indeed, in document processing applications omission errors are deemed to be less serious than commission errors, which can lead to totally erroneous storing of data in information systems. As regards the interpretation of the learning method results, it is noteworthy that the first-order learning method proposed in this article yields layout/logical structure models that can be easily checked and interpreted with visualization techniques.

The purpose of the present article is to illustrate problems in the application of machine learning techniques to various document processing steps performed by the system WISDOM++. Such steps are described in the next section, which clarifies the learning problems, the learning tasks, the preprocessing of original data (i.e., document images), as well as the two main document structures (i.e., layout and logical) involved in the various processing steps. In Section 3, some representation and learning issues concerning the block classification problem are discussed, the solutions adopted in WISDOM++ are reported, and the results of an empirical evaluation are commented. Section 4 is devoted to the presentation of the first-order learning system used for both document classification and document understanding. The system, named *INDUBI/CSL* (Malerba et al., 1997c), has been extended in order to handle both symbolic and numeric attributes and relations, and presents some distinguishing features with respect to other well-known first-order learning systems. Some experimental results on document classification and understanding are commented in Section 5. The article concludes with a brief discussion on results achieved in this work and unresolved problems.

2. Functional architecture of WISDOM++

The tasks performed by WISDOM++ (www.di.uniba.it/∼malerba/wisdom++/) and the intermediate results produced at each processing step are reported in figure 3.

Initially, each page is scanned with a resolution of 300 dpi and thresholded into a binary image. The bitmap of an A4-sized page takes 2,496 $\times$ 3,500 $=$ 1,092,000 bytes and is stored in TIFF format. Actually, WISDOM++ can manage multi-page documents, each of which is a *sequence* of pages. The definition of the right sequence is the responsibility of the user, since the scanner is able to scan a single page at time. Pages of multi-page documents are processed independently of each other in all steps, therefore the document processing flow is described for single pages only.

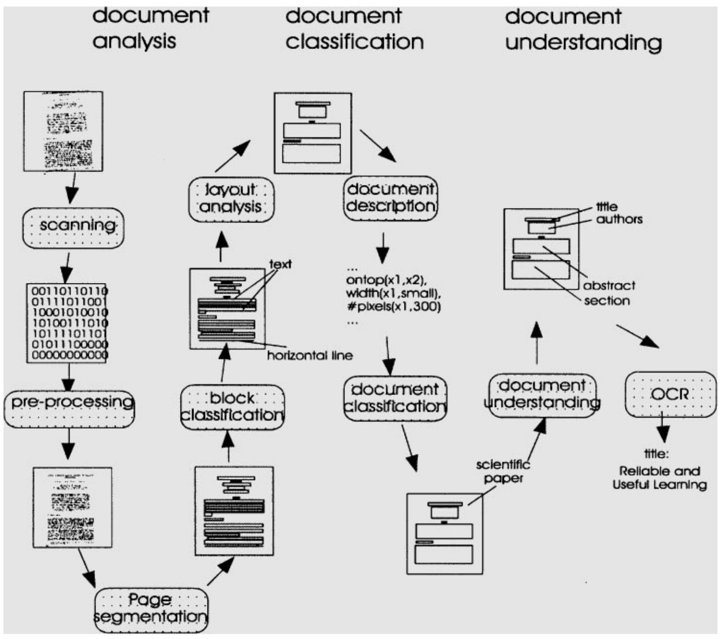


Figure 3. Functional architecture of WISDOM++.

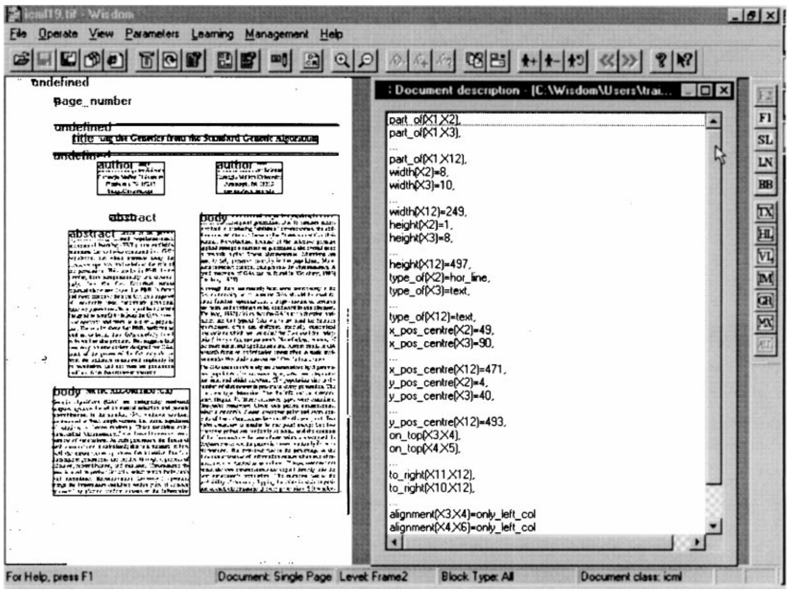
The document analysis process is quite complex and includes:

1. *Preprocessing*, that is, the evaluation of the skew angle, the rotation of the document, and the computation of a spread factor. In particular, the skew angle is estimated by analyzing the *horizontal projection profile*, that is, a histogram reporting the total number of black pixels for each row of the bitmap. The histogram shows sharply-rising peaks with the base equal to the character height when text lines span horizontally, while it is characterized by smooth slopes when a large skew angle is present. This observation can be mathematically expressed by a real-valued function, called the *alignment measure*, that returns the mean square deviation of the histogram for each orientation angle θ. Thus, finding the actual skew angle is reformulated as the problem of locating the global maximum value of the alignment measure. Since this measure is not smooth enough to apply usual gradient ascent techniques, the system adopts some peak-finding heuristics (Altamura et al., 1999). Once the skew angle is estimated, the document is rotated. The loop "skew estimation—document rotation" terminates when the estimated skew angle is zero. At that point WISDOM++ estimates the *spread factor* of the document image as the ratio of the mean distance between peaks and the peak width. Such a ratio is greater than (lower than) 1.0 for simple (complex) documents.
2. *Segmentation*, that is, the identification of rectangular blocks enclosing content portions. The segmentation is performed on a document image with a resolution of only 75 dpi (*reduced* document image), which is deemed a reasonable trade-off between the accuracy and the speed of the segmentation process. The page is segmented by means of a fast technique, named *Run Length Smoothing Algorithm* (RLSA) (Wong et al., 1982), that applies four operators to the document image: 1) Horizontal smoothing with a threshold C_h; 2) Vertical smoothing with a threshold C_v; 3) Logical AND of the two smoothed images; 4) additional horizontal smoothing with another threshold C_a. The novelty of WISDOM++ is that the smoothing parameters C_v and C_a are defined on the grounds of the spread factor.
3. *Block classification*, which aims at distinguishing blocks enclosing text from blocks enclosing graphics (pictures, drawings and horizontal/vertical lines).
4. *Layout analysis*, that is, the perceptual organization process that detects structures among blocks. The result is a hierarchy of abstract representations of the document image, called *layout structure*. The leaves of the layout tree (lowest level of the abstraction hierarchy) are the blocks, while the root represents the set of pages of the whole document. A page may include several layout components, called *frames*, which are still rectangular areas corresponding to groups of blocks. An ideal layout analysis process should segment a page into a set of frames, such that each frame can be associated with a distinct semantic label (e.g., title and author of a scientific paper). WISDOM++ extracts the layout structure by means of a knowledge-based approach: Generic knowledge and rules on typesetting conventions are used in order to group basic blocks together (Malerba et al., 1995). The layout hierarchy has six levels: basic blocks, lines, set of lines, frame1, frame2, and pages.

While the layout structure associates the content of a document with a hierarchy of layout objects, such as blocks, frames and pages, the *logical structure* of the document associates the content with a hierarchy of *logical objects*, such as sender/receiver of a business letter, title/authors of a scientific article, and so on. The problem of finding the logical structure of a document can be reformulated as the problem of defining a mapping from the layout structure into the logical one. In WISDOM++ this mapping is limited to the association of a page with a document class (document classification) and the association of page layout components with basic logical components (document understanding). The mapping is built by matching the document description with *models* of classes of documents and against models of the logical components of interest for that class.

The descriptions of both layout structures and models are given in a first-order language, where unary function symbols, called *attributes*, are used to describe properties of a single layout component (e.g., height and length), while binary predicate and function symbols, called *relations*, are used to express interrelationships among layout components (e.g., contain, on-top, and so on). Attributes and relations can be both symbolic (i.e., categorical and ordinal) and numeric. An example of description in first-order language of a page layout is reported in figure 4.

Models are represented as rules. Typically, such rules are handcoded for particular kinds of documents (Nagy et al., 1992), requiring much human tuning and effort. WISDOM++ uses rules that are automatically generated from a set of training examples for which the user-trainer has already defined the correct class and has specified the layout components with



```
part_of(X1,X2),
part_of(X1,X3),
...
part_of(X1,X12),
width(X2)=8,
width(X3)=10,
...
width(X12)=249,
height(X2)=1,
height(X3)=8,
...
height(X12)=497,
type_of(X2)=hor_line,
type_of(X3)=text,
...
type_of(X12)=text,
x_pos_centre(X2)=49,
x_pos_centre(X3)=90,
...
x_pos_centre(X12)=471,
y_pos_centre(X2)=4,
y_pos_centre(X3)=40,
...
y_pos_centre(X12)=493,
on_top(X3,X4),
on_top(X4,X5),
...
to_right(X11,X12),
to_right(X10,X12),
...
alignment(X3,X4)=only_left_col
alignment(X4,X6)=only_left_col
```

For Help, press F1 | Document: Single Page | Level: Frame2 | Block Type: All | Document class: icml

Figure 4. Frame2 layout level of a document processed by WISDOM++ (left). The level of the layout hierarchy to be displayed is chosen by clicking on the radio buttons F2 (frame2), F1 (frame1), SL (set of lines), LN (lines) and BB (basic blocks). The document has been classified as an ICML95 paper (see status bar) and its logical structure has been understood (see labels associated to the layout components). The first-order logical description of the page layout is reported on the right window. Distinct constants (X2, . . . , X12) denote distinct layout components. The constant X1 denotes the whole page.

a logical meaning (logical components). The first-order learning system used to generate such rules is INDUBI/CSL (Malerba et al., 1997c).

Finally, WISDOM++ allows the user to set up the text extraction process by selecting the logical components to which the OCR has to be applied. Then the system generates an HTML/XML version of the original document. This contains both text read by the OCR and pictures extracted from the original bitmap and converted into the GIF format. Text and images are spatially arranged so that HTML/XML reconstruction of the document is as faithful as possible to the original bitmap. Moreover, the XML format maintains information extracted during the document understanding phase, since the Document Type Definition (DTD) is specialized for each class of documents to represent the specific logical structure.

3. Decision tree learning for block classification

Page segmentation defines blocks that may contain either textual or graphical information. In order to facilitate subsequent document processing steps, it is important to label these blocks according to the type of content. Labels used in WISDOM++ are text block, horizontal line, vertical line, picture (i.e., halftone image) and graphics (e.g., line drawing). Each block can be associated with only one label, so that the labeling problem can be reformulated as a classification problem where labels correspond to classes.

A method for document block classification was proposed by Wong et al. (1982). The basic features used for classifying blocks are four:

1. The height of each block.
2. The eccentricity of the rectangle surrounding the block.
3. The ratio of the number of black pixels to the area of the surrounding rectangle.
4. The mean horizontal length of the black runs of the original block image.

Additional features concerning the texture of the blocks were proposed by Wang and Srihari (1989). A common aspect of these works is the use of linear discriminant functions as classifiers. An exception is the work by Fisher et al. (1990), who resort to a rule-based classification method, where rules are hand-coded.

In WISDOM++, block classification is performed by means of a decision tree automatically built from a set of training examples (blocks) of the five classes. The choice of a “tree-based” method instead of the most common generalized linear models is due to its inherent flexibility, since decision trees can easily handle complicated interactions among features and give results that are simple to interpret.

In a preliminary study, a decision tree has been induced from a set of 5,473 examples of pre-classified blocks obtained from 53 documents of various kinds.[1] Two different decision tree learning systems have been considered: C4.5 (Quinlan, 1993) and ITI (Utgoff, 1994). The former is a *batch* learner, that is it cannot change the decision tree when some blocks are misclassified, unless a new tree is generated from scratch using an extended training set. On the contrary, ITI allows *incremental* induction of decision trees, since it can revise the current decision tree, if necessary, in response to each newly observed training instance. Currently, ITI is the only incremental system able to handle numerical attributes as those used to describe the blocks.

Table 1. Experimental results for a ten-fold cross-validation performed on the data set of 5,473 examples. Both systems are allowed to prune the induced trees using their default pruning method (error-based pruning for C4.5 and MDL pruning for ITI).

	C4.5	ITI
Average no. of leaves	89.8	92.4
Predictive accuracy	96.8	96.88

Actually, ITI can operate in three different ways. In the *batch* mode, it works in a way similar to C4.5. In the *normal* operation mode, it first updates the frequency counts associated to each node of the tree as soon as a new instance is received. Then it restructures the decision tree according to the updated frequency counts. When working in this mode, ITI builds trees with almost the same number of leaves and the same predictive accuracy of those induced with C4.5 (see Table 1). In the *error-correction* mode, frequency counts are updated only in case of misclassification of the new instance. The main difference between the two incremental modes is that the normal operation mode guarantees to build the same decision tree independently of the order in which examples are presented, while the error-correction mode does not.

From the practical point of view, the main problem we observed is that ITI creates large files (more than 10 Mb) when trained on the data set of 5,473 instances. The reason of this space inefficiency is due to the need of storing frequency counts of training examples in each node of the induced tree. This inefficiency can be contained (about 1.5 Mb) when the system operates in the error-correction mode, since frequency counts are updated only in case of misclassification.

Similar results have also been obtained in a recent empirical evaluation performed on a set of 112 documents (Altamura et al., 1999). In this case, the training set includes 9,429 examples and the decision tree built in the batch mode takes more than 24 Mb, while the decision tree obtained with the error-correction mode requires 982 Kb. Nevertheless, the difference between the predictive accuracy in the batch mode and in the error correction mode is less than 0.2% if estimated on an independent test set of 3,176 examples. The numerical features used by WISDOM++ to describe each block are reported in Table 2. Two features, namely width and F3, are never tested in the decision tree built with the pure error-correction mode. The former is especially useful to recognize horizontal/vertical lines, but the decision tree learning system prefers to test on the equally informative feature eccentricity. The latter is excluded since it characterizes extra-long runs of black pixels, as those found in newspaper headlines, while documents considered in this experiment have few occurrences of large text blocks.

Two new functions have been added to WISDOM++: The interactive correction of the results of the block classification, and the updating of the decision tree produced by ITI. The preference for ITI is due to the possibility of on-line training the document processing system: When users are dissatisfied with the classification performed by the induced decision tree, they can ask the system to revise the classifier without starting from scratch. In this way, some blocks (e.g., the logo of a business letter) can be considered text for some users and graphics for others.

Table 2. Features used to describe segmented blocks.

Feature	Description
Height	Height of the reduced block image
Width	Width of the reduced block image
Area	Area of the reduced block image (Height $*$ Width)
Eccen	Eccentricity of the reduced block image (Width/Height)
Blackpix	Total number of black pixels in the reduced block image
Bw_trans	Total number of black-white transitions in all rows of the reduced block image
Pblack	Percentage of black pixels in the reduced block image (Blackpix/Area)
Mean_tr	Average number of black pixels per black-white transition (Blackpix/Bw_trans)
F1	Short run emphasis
F2	Long run emphasis
F3	Extra long run emphasis

4. First-order learning for document classification and understanding

In a previous work on the problem of processing paper documents, a multistrategy learning approach was proposed to learn a set of rules for both document classification and document understanding (Esposito et al., 1994). In particular, a learning methodology that integrates a parametric and a conceptual learning method was adopted. In fact, the application required the management of first-order representations with both numeric and symbolic data, while the conceptual learning method was able to deal exclusively with symbolic data. The integrated learning methodology committed the management of all numeric attributes to the parametric method and the handling of symbolic attributes and relations to the first-order conceptual method. The main limitation of this approach is that the parametric method can manage only zero-order (i.e., feature vector) representations. Thus numeric attributes concerns only “global” properties of the whole document, while “local” properties of a component of the document (e.g., height, width and position) have to be discretized before starting the conceptual learning process.

Preliminary results obtained with the integrated approach were encouraging, but not totally satisfying. This prompted the investigation of an extension of the first-order learning method implemented in INDUBI/CSL (Malerba et al., 1997c), a general-purpose learning system used in this application.

4.1. The learning algorithm

The learning problem solved by INDUBI/CSL is that of inducing a set of hypotheses $H_1, H_2, \ldots, H_r$, from a set E of training examples. Each hypothesis H_i is the description of a concept C_i.

The representation language adopted by the system has two distinct forms of *literals*:

$$f(t_1, \ldots, t_n) = \textit{Value}\ (\textit{simple literal}) \text{ and } f(t_1, \ldots, t_n) \in \textit{Range}\ (\textit{set literal})$$

where f is an n-ary function symbol, called *descriptor*, t_i's can be either variable or constant terms, *Value* is the value taken by f when applied to $t_1, \ldots, t_n$, and *Range* is a set of possible values taken by f. Some examples of literals are the following: $height(x2) \in [1.1..2.1]$, $color(x1) = red$, $distance(x1, x2) \in [0.0..1.0]$. Literals can be combined to form *definite clauses*:

$$L_0 \leftarrow L_1, L_2, \ldots, L_m$$

where the simple literal L_0 is called *head* of the clause, while the conjunction of simple or set literals $L_1, L_2, \ldots, L_m$ is named *body*. Definite clauses of interest for classification problems satisfy two different constraints: *Linkedness* (Helft, 1987) and *range-restrictedness* (De Raedt, 1992).

Each training example is represented as a single ground, linked and range-restricted definite clause. On the contrary, a hypothesis H is a set of linked, range-restricted definite clauses, called *rule*, such that all clauses have the same head and no constant terms. Permitted literals in H can be either single-valued or range-valued.

At the high level INDUBI/CSL implements a *separate-and-conquer* (or *sequential covering* (Mitchell, 1997)) search strategy to generate a rule. With reference to figure 5, the *separate* stage corresponds to the external loop that checks for the completeness of the current rule.[2] If this check fails, the search for a new consistent clause is begun.[3] The search space for the separate stage is the set of rules, while the search space of the *conquer* stage is the set of clauses. The conquer stage performs a general-to-specific beam-search to construct a new consistent, linked and range-restricted clause. The separate-and-conquer search strategy is adopted in other well-known learning systems, such as FOIL (Quinlan and Cameron-Jones, 1993). On the other hand, INDUBI/CSL bases the conquer phase on the concept of *seed* example, whose aim is that of guiding the generalization process. Indeed, if e^+ is a positive example to be explained by a hypothesis H, then H should contain at least one clause C that generalizes e^+. Such a clause can be obtained from e^+ by applying two distinct operators: *Dropping-condition* (or *dropping-literal*) and *turning-constants-into-variables*.

```
1.  Separate-and-Conquer(Examples, Concept, M, N, P, ConsCrit, SpecCrit):
2.      $E^+$ := set of positive Examples of Concept;
3.      $E^-$ := set of negative Examples of Concept;
4.      $E_0^+$ := $E^+$;
5.      LearnedRule := $\emptyset$;
6.      while $E^+ \neq \emptyset$ do
7.          select the seed $e^+$ from $E^+$
8.          Cons := Beam-Search-for-Consistent-Clauses($e^+$, $E^+$, $E^-$, $E_0^+$, M, N, P, SpecCrit);
9.          Best := Select-Best-Cons(Cons, ConsCrit);
10.         LearnedRule := LearnedRule $\cup$ {Best};
11.         $E^+$ := $E^+ \setminus$covers(Best, $E^+$);
12.     endwhile;
13. return(LearnedRule).
```

Figure 5. Separate-and-conquer search strategy.

```
1.  **Beam-Search-for-Consistent-Clauses**($e^+$, $E^+$, $E^-$, $E_0^+$, M, N, P, SpecCrit):
2.      Spec := {Turning-Constants-into-Variables(head($e^+$)←, ∅, $e^+$)};
3.      Cons := ∅;
4.      *while* Spec ≠ ∅ and |Cons| <M *do*
5.          OldSpec := Spec;
6.          Spec := ∅;
7.          *foreach* clause G in OldSpec *do*
8.              Spec := Spec ∪ **Specialize-G-by-Adding-a-Literal**(G, $e^+$, N, $E^+$, $E^-$);
9.          *endforeach*;
10.         Cons := Cons ∪ {consistent(Spec) ∩ range-restricted(Spec)};
11.         Spec := Select-Best-Spec(Spec, P, SpecCrit, $E^+$, $E^-$, $E_0^+$);
12.     *endwhile*
13. *return*(Cons).
```

Figure 6. Beam search for consistent clauses that cover e^+.

To sum up, INDUBI/CSL starts with a seed example e^+ and generates a set *Cons* of clauses that are consistent and cover e^+. The best clause is selected from *Cons* according to a user-defined preference criterion (*ConsCrit*), and the positive examples covered by such a clause are removed from the set of examples to be covered. If there are still some positive examples to be covered, a new seed will be selected in the next iteration.

In the beam search (see figure 6), INDUBI/CSL searches for M, if any, distinct clauses that are consistent and range-restricted. Instead of searching *bottom-up* by turning constants of e^+ into variables and then progressively dropping literals in the body, the system proceeds *top-down* by specializing the most general hypothesis that covers e^+. Such a hypothesis is a definite clause with empty body,

$$f(X_1, \ldots, X_n) = \mathit{Value} \leftarrow$$

which is obtained by applying the operator turning-constants-into-variables to the head of e^+. The specialization operator applied in this search is named *adding-literal* and it is the dual of the generalization operator dropping-literal. During this top-down search, INDUBI/CSL ranks clauses according to a user-defined preference criterion (*SpecCrit*). Consistent clauses are copied into *Cons* and the first P clauses are selected for the next specialization step. P is the *beam* of the search.

The specialization operator has been extended in order to handle both numeric and symbolic descriptors (Malerba et al., 1997a). This extension satisfies the following conditions:

1. The specialized clause has to be linked.
2. In the case of numerical descriptors, the specialization of a clause G is obtained by adding literals of the type $f(X_1, \ldots, X_n) \in [a..b]$, where the interval $[a..b]$ is defined according to G itself (*on-line local discretization*).
3. The heuristic function used to choose among different intervals $[a..b]$ should satisfy a property that reduces the computational complexity of the operator.
4. The specialized clause should cover the seed example e^+.

```
1.  Specialize-G-by-Adding-a-Literal(G, $e^+$, N, $E^+$, $E^-$):
2.      List-linked := $\emptyset$;
3.      foreach literal Lit in $e^+$ do
4.          [f($X_1$,... ,$X_n$)=Value] := Turning-Constants-into-Variables(Lit, G, $e^+$);
5.          if [f($X_1$,...,$X_n$)=Value] is linked to G then
6.              if the descriptor $f$ is numeric then
7.                  Set-literal := Determine-Range([f($X_1$, ...,$X_n$)=Value], $e^+$, $E^+$, $E^-$);
8.                  if Set-literal$\neq$nil then add Set-literal to List-linked endif
9.              else
10.                 if [f($X_1$,... ,$X_n$)=Value] is not in G then
11.                     add [f($X_1$,... ,$X_n$)=Value] to List-linked
12.                 endif
13.             endif
14.     endforeach
15.     LL := Select-Best-List-linked(List-linked, N)
16.     GSpec := $\emptyset$;
17.     foreach literal L in LL do
18.         GSpec := GSpec $\cup$ Specialize-G-by-Adding-L(G, L);
19.     endif
20. return (GSpec).
```

Figure 7. Specialization operator for definite clauses.

4.2. The specialization operator

In order to specialize a clause G, INDUBI/CSL has to choose some literals to be added. All candidate literals are generalizations of literals in the seed, e^+, obtained by turning distinct constants into distinct variables. Both numeric and symbolic data are handled in the same way (see figure 7). The only difference is that numeric literals already present in G can be reconsidered later on. In this case, the best (sub-)interval is recomputed, since it may be influenced by the addition of further literals. For the specialization process, only a subset of N linked literals is actually considered: the selection of literals is based on the cost associated to the descriptor f, so that the user can express a preference for some literals.

In order to compute the interval for numeric literals (see figure 8), the system builds a table associated to the term $f(X_1, \ldots, X_n)$ by matching the specialized clause

$$G' : G,\ f(X_1, \ldots, X_n) \in [-\infty .. +\infty]$$

with positive and negative examples. Each example produces as many entries as the number of unifiers. The table, initially empty, contains pairs $\langle Value, Class \rangle$, where *Class* can be either $+$ or $-$ according to the sign of the example e from which *Value* is taken, while *Value* is determined by considering the literal of e that unifies with $f(X_1, \ldots, X_n) \in [-\infty .. +\infty]$. Then the problem is finding the interval that best discriminates positive from negative examples. Any threshold value α lying between two consecutive distinct values defines two disjoint intervals: The left interval $[l_1, l_2]$ and the right interval $[r_1, r_2]$. The lower bound l_1 of the left interval is the smallest value in the table with a $+$ sign, while the upper bound

```
1.  Determine-Range([f(X_1,...,X_n)=SeedValue], e^+, E^+, E^-)
2.      initialize table T[f(X_1,...,X_n)]
3.      foreach example e in E^+ do
4.          foreach substitution $\theta$ s.t. G,f(X_1,...,X_n) $\in [-\infty..+\infty]$ covers e do
5.              select the literal [f(X_1,...,X_n)=Value]$\theta$ of e
6.              add the tuple <Value,+> to the table T[f(X_1,...,X_n)];
7.          endforeach
8.      endforeach
9.      foreach example e in E^- do
10.         foreach substitution $\theta$ s.t. G,f(X_1,...,X_n) $\in [-\infty..+\infty]$ covers e do
11.             select the literal [f(X_1,...,X_n)=Value]$\theta$ of e
12.             add the tuple $\langle$Value,-$\rangle$ to the table T[f(X_1,...,X_n)];
13.         endforeach
14.     endforeach
15.     sort table T[f(X_1,...,X_n)] on the Value field
16.     Cut := Determine-All-Cut-Points(T[f(X_1,...,X_n)])
17.     MaxIG := $-\infty$
18.     foreach cut-point $\alpha$ in Cut do
19.         determine $[l_1,l_2]$, $[r_1,r_2]$ with $l_2 < \alpha < r_1$
20.         if SeedValue$\in [l_1,l_2]$ then AdmissibleInterval := $[l_1,l_2]$
21.             else AdmissibleInterval := $[r_1,r_2]$ endif
22.         IG := Information-Gain (T[f(X_1,...,X_n)], AdmissibleInterval)
23.         if IG>MaxIG then
24.             BestInterval := AdmissibleInterval
25.             MaxIG := IG
26.         endif
27.     endforeach
28.     if MaxIG$\neq -\infty$ then return(f(X_1,...,X_n) $\in$ BestInterval)
29.         else return(nil) endif
```

Figure 8. Choice of the best interval for numeric descriptors.

l_2 is the largest value in the table that does not exceed the threshold α. On the contrary, the lower bound r_1 of the right interval is the smallest value in the table that exceeds α, while the upper bound r_2 is the largest value with a $+$ sign. When one of the two intervals contains no positive value, then it is set to *undefined.* At least one of the two intervals must be defined, since the table contains at least one entry $\langle$*SeedValue*, $+\rangle$ for the value taken by $f(X_1, \ldots, X_n)$ in the seed. Not all defined intervals are to be considered, since the specialized clause

$$G'' : G,\ f(X_1, \ldots, X_n) \in \mathit{Range}$$

for a given *Range* might no longer cover the seed example e^+. Those intervals that include the *SeedValue* are said to be *admissible,* because they guarantee that the corresponding specializations still cover e^+.

The best admissible interval is selected according to an information-theoretic heuristics. By looking at the table as a source of messages labeled $+$ and $-$, the expected information on the class membership conveyed from a randomly selected message is:

$$info(n^+, n^-) = -\frac{n^+}{n^+ + n^-} \log_2 \frac{n^+}{n^+ + n^-} - \frac{n^-}{n^+ + n^-} \log_2 \frac{n^-}{n^+ + n^-}$$

where n^+ and n^- are the number of values in the table with a positive and a negative sign, respectively. If we partition the table into two subsets, S_1 and S_2, the former containing $n_1^+ + n_1^-$ values falling within an admissible interval and the latter containing the remaining values, the information provided by S_1 will be close to zero when almost all cases have the same $+$ or $-$ sign. Although the information prefers partitions that cover a large number of cases of a single class and a few cases of other classes, we must bias such a preference towards intervals with a high number of positive cases, as well. The following *weighted entropy*:

$$E(n_1^+, n_1^-) = \frac{n_1^-}{n_1^+} info(n_1^+, n_1^-)$$

penalizes those admissible intervals with a low percentage of positive cases. A heuristic criterion is that of choosing the admissible interval that minimizes the weighted entropy. It differs from that adopted in C4.5, where the entropy is not weighted, and in FOIL where only the information content of the positive class is considered.

As a concrete illustration of the procedure *determine-range*, consider the table below.

Value	0.5	0.7	0.9	1.0	1.5	1.5	1.5	1.7	2.5	2.5
Sign	$+$	$-$	$-$	$-$	$-$	$-$	$+$	$+$	$-$	$+$

There are four possible cut points that generate the following intervals:

α	0.60	1.25	1.60	2.10
$[l_1, l_2]$	[0.50..0.50]	[0.50..1.00]	[0.50..1.50]	[0.50..1.70]
$[r_1, r_2]$	[0.70..2.50]	[1.50..2.50]	[1.70..2.50]	[2.50..2.50]

Let us suppose that *Seed_value* equals 1.50. Then only those intervals including 1.50 are admissible. The weighted entropy for each of them is

Admissible interval	[0.70..2.50]	[1.50..2.50]	[0.50..1.50]	[0.50..1.70]
Entropy E	1.836592	1.000000	2.157801	1.590723

Thus, the best interval is the second one, with weighted entropy equal to 1.0.

Note that cut points 0.80 and 0.95 have not been considered. Indeed, only those between two consecutive distinct values with a different sign (*boundary points*) are considered. This choice is due to the following theorem:

Theorem 1. *If a cut-point α minimizes the measure $E(n_1^+, n_1^-)$, then α is a boundary point.*

The proof is reported in Appendix A. This result helps to reduce computational complexity of the procedure *determine-range* by considering only boundary points. Actually, the theorem above is similar to that proved by Fayyad and Irani (1992) for a different measure, namely the "unweighted" class information entropy computed in some decision tree learning systems.

4.3. Handling numerical data in first-order learning: other approaches

The first attempt to deal with continuous-valued attributes in first-order systems that learn classification was made by Bergadano and Bisio (1988), who proposed a method to automatically set some parameters of predicate "schemes" with a parametric semantics. Later on, a two-step approach was implemented in the system ML-SMART (Botta and Giordana, 1991): First, a tentative numerical parameter is learned, and then a standard genetic algorithm is applied to refine the numerical knowledge. On the contrary, the system Rigel (Gemello et al., 1991) discretizes continuous data by applying a generalization operator called *consistent extending reference rule*, which extends the reference of a selector, that is the set of values taken by a function f. Values are added only if this increases the number of covered positive examples without covering negative ones. A different approach was proposed by Esposito et al. (1993) who combined a discriminant analysis technique for linear classification with a first-order learning method, so that the numerical information is handled by linear classifiers, while the symbolic attributes and relations are used by the first-order learning system. A common characteristic of all these approaches is that they have been conceived for systems that can learn only rules with nullary predicates in the head, that is with predicates corresponding to propositional classes.

Dzeroski and Bratko (1996) proposed transforming first-order representations into propositional form, in order to handle real numbers by means of techniques already tested in decision tree learning systems. Nevertheless, the transformation algorithm is applicable only when the background relations are *determinate* (Lavrac and Dzeroski, 1994).

A different approach has been adopted in FOIL 6.2 (Quinlan and Cameron-Jones, 1993). The system automatically produces comparative literals of type $V_i > k$, $V_i \leq k$, $V_i > V_j$, $V_i \leq V_j$, where V_i and V_j are numerical variables already present in other non-comparative literals and k is a numerical threshold. The selection of the threshold is based on an information-theoretic measure, which is different from that adopted in our system. Indeed, FOIL's specialization operator does not guarantee it will cover a specific positive example, the seed. Other differences between the two systems concern the top-down learning process (beam-search, seed-driven vs. hill-climbing, information-gain-driven), and the stopping criterion (at least M consistent hypotheses found vs. minimum description length).

Much related work can also be found in other contexts, such as qualitative and relational regression in inductive logic programming, and learning numerical constraints in inductive constraint logic programming. An updated review can be found in Lavrac et al. (1996).

5. Experimental results on document classification and understanding

In order to test the efficiency and the effectiveness of the first-order learning algorithm, an experiment on the domains of document classification and understanding has been organized.

INDUBI/CSL has been applied to the problems of classifying and understanding a set of 112 real, single-page documents distributed as follows: Twenty-eight articles of ICML'95, thirty articles of ISMIS'94,[4] thirty-four articles of IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI),[5] and twenty documents of different type or published on other proceedings, transactions and journals (Reject). All documents of the first three classes are first-pages of articles. For ICML'95 papers, five logical components are of interest, namely page number, title, author, abstract and body of the paper. For ISMIS'94 papers the following logical components are considered: Title, author, abstract and body of the paper. Finally, for TPAMI papers only five logical components are defined, namely running head, page number, title, abstract, and body of the paper. For the remaining documents it was not possible to define a set of logical components because of the high variability. Thus there are four learning problems to be solved: Learning to classify documents in the classes ICML'95, ISMIS'94, TPAMI and Reject, and learning to identify logical components in ICML'95/ ISMIS'94/TPAMI papers. In document understanding problems each document generates as many training examples as the number of layout components (in this experiment, only components at the *frame2* level are considered). Each training example is represented as a definite ground clause, where different constants represent distinct components of a page layout. The choice of a representation language for the description of the layout of each document is very important. In previous experiments we used only symbolic descriptors by discretizing numeric attributes such as height, width and position of a block (see Table 3). Since the current release of INDUBI/CSL is able to handle numerical descriptors as well, we decided to organize an experiment to test the improvement of the generated rules in terms of accuracy, learning time and simplicity.

The experimental procedure followed is ten-fold cross validation. At each trial three statistics are collected: Number of errors on the test cases, number of generated clauses, and learning time. Results for the document classification are reported in Table 4. The entries labelled 'mixed' refer to symbolic/numeric representation and on-line discretization.

In the document classification problem, the average number of errors for mixed data is significantly lower than the average number of errors for symbolic data. Indeed, the p-value returned by the non-parametric Wilcoxon signed-ranks test (Orkin and Drogin, 1990) is 0.0144. Moreover, the introduction of numerical descriptors simplifies the classification rules, although the learning time is doubled (time is in prime minutes and refers to a SUN 10). Two examples of rules learned by the embedded system are reported below. It is worthwhile to observe that such rules capture some spatial relationships between layout components, thus confirming the importance of adopting a first-order representation:

$$\begin{aligned} \mathit{class}(X) = \mathit{icml} \leftarrow\ & \mathit{part\_of}(X,Y),\ \mathit{on\_top}(W,Z),\ \mathit{on\_top}(Y,U),\ \mathit{to\_right}(Y,V),\\ & \mathit{alignment}(Z,U) = \mathit{only\_left\_col},\\ & \mathit{alignment}(V,Y) = \mathit{only\_middle\_col} \end{aligned}$$

Table 3. Descriptors used in the first-order representation of the layout components.

Descriptor name	Definition
width(block)	Integer domain (1..640) in numerical descriptions
	Linear domain in symbolic descriptions: very_very_small, very_small, small, medium_small, medium, medium_large, large, very_large, very_very_large
height(block)	Integer domain (1..875) in numerical descriptions
	Linear domain in symbolic descriptions: very_very_small, very_small, small, medium_small, medium, medium_large, large, very_large, very_very_large
x_pos_centre(block)	Integer domain
y_pos_centre(block)	Integer domain
position(block)	Nominal domain (used in symbolic descriptions in the place of *x_pos_centre* and *y_pos_centre*): top_left, top, top_right, left, center, right, bottom_left, bottom, bottom_right
type_of(block)	Nominal domain: text, hor_line, image, ver_line, graphic, mixed
part_of(block1,block2)	Boolean domain: true if block1 contains block2
on_top(block1,block2)	Boolean domain: true if block1 is above block2
to_right(block1,block2)	Boolean domain: true if block2 is to the right of block1
alignment(block1,block2)	Nominal domain: only_left_col, only_right_col, only_middle_col, only_upper_row, only_lower_row, only_middle_row

Table 4. Experimental results for document classification.

	Av. no. errors		No. clauses		Av. learning time	
Class	Symbolic	Mixed	Symbolic	Mixed	Symbolic	Mixed
ICML'95	1.4	0.8	4.8	2.0	4 : 42	9 : 54
ISMIS'94	1.3	0.2	5.7	1.0	5 : 30	9 : 18
TPAMI	1.0	0.7	4.4	1.9	5 : 06	10 : 06
TOTAL	3.7	1.7	14.9	4.9	15 : 36	29 : 30

$$\begin{aligned} class(X) = icml \leftarrow\, & part\_of(X,Y),\ x\_pos\_centre(Y) \in [301..557],\\ & y\_pos\_centre(Y) \in [25..190],\\ & on\_top(V,Y),\ on\_top(Z,V),\ on\_top(W,Y) \end{aligned}$$

Results for the three document understanding problems are reported in Table 5. The set of rules learned by INDUBI/CSL for the ICML95 documents in one experiment are

Table 5. Experimental results for document understanding.

	Av. no. errors		No. clauses		Av. learning time	
Class	Symbolic	Mixed	Symbolic	Mixed	Symbolic	Mixed
ICML'95	6.6	3.2	31.6	8.4	44 : 30	26 : 00
ISMIS'94	6.3	3.2	43.8	10.8	50 : 36	25 : 24
TPAMI	9.2	2.0	39.5	9.0	30 : 24	32 : 00

reported below:

$$\begin{array}{lll}
\textit{logic\_type}(X) = \textit{page\_number} & \leftarrow & \textit{width}(X) \in [8..16], \textit{height}(X) \in [7..8] \\
\textit{logic\_type}(X) = \textit{title} & \leftarrow & \textit{height}(X) \in [13..31], \\
 & & \textit{x\_pos\_centre}(X) \in [280..348] \\
\textit{logic\_type}(X) = \textit{author} & \leftarrow & \textit{height}(X) \in [42..79], \\
 & & \textit{y\_pos\_centre}(X) \in [173..279] \\
\textit{logic\_type}(X) = \textit{abstract} & \leftarrow & \textit{y\_pos\_centre}(X) \in [256..526], \\
 & & \textit{on\_top}(Y, X), \textit{to\_right}(X, Z) \\
\textit{logic\_type}(X) = \textit{abstract} & \leftarrow & \textit{x\_pos\_centre}(X) \in [147..218], \\
 & & \textit{on\_top}(X, Y), \textit{to\_right}(Y, Z) \\
\textit{logic\_type}(X) = \textit{body} & \leftarrow & \textit{width}(X) \in [242..255], \textit{type\_of}(X) = \textit{text} \\
\textit{logic\_type}(X) = \textit{body} & \leftarrow & \textit{x\_pos\_centre}(X) \in [368..477], \\
 & & \textit{on\_top}(Z, X), \textit{to\_right}(Y, Z) \\
\textit{logic\_type}(X) = \textit{body} & \leftarrow & \textit{width}(X) \in [78..136], \textit{type\_of}(X) = \textit{text}, \\
 & & \textit{on\_top}(X, Y), \textit{to\_right}(Y, Z) \\
\textit{logic\_type}(X) = \textit{body} & \leftarrow & \textit{width}(X) \in [237..255], \\
 & & \textit{alignment}(Y, X) = \textit{only\_right\_col} \\
\textit{logic\_type}(X) = \textit{body} & \leftarrow & \textit{height}(X) \in [422..519]
\end{array}$$

As expected, the choice of the representation language for the page layout is a critical factor in the document understanding. Indeed, satisfactory error rates have been obtained only with numeric/symbolic representations. For all classes the average number of errors for mixed data is significantly lower than the average number of errors for symbolic data (the worst p-value of the Wilcoxon test is 0.0072). Once again, the introduction of numerical descriptors simplifies significantly the classification rules and decreases the learning time for two classes.

5.1. *Conclusions*

Experimental results reported above prove that machine learning techniques can be helpful to acquiring specific knowledge for intelligent processing of printed documents. In particular, two decision tree learning systems have been successfully applied to the problem of classifying blocks defined by the segmentation algorithm, while a first-order rule learning system has been effectively used for document classification and understanding.

As regards the block classification problem, three different approaches to the top-down induction of decision trees have been investigated: Batch, normal incremental, and error-correction incremental. The main advantage of the two incremental approaches is that decision trees can be properly revised as new training examples are available. Revising a decision tree is generally less expensive than learning a new decision tree from an extended training set. This is an important aspect to consider when real-time user interaction is a requirement of document processing applications. On the other hand, a pure incremental learning algorithm can require a huge amount of memory to store all necessary information, thus making this approach unfeasible. In our work, the error-correction incremental approach has proved to be a reasonable compromise between predictive accuracy and space complexity. As future work, the problem of efficiently handling numerical features in the incremental induction of decision trees will be further investigated.

The main issue addressed for the document classification and understanding problems is the representation of the layout structures. In the paper, the most abstract level of the layout structure of documents has been described by means of a first-order logic formalism, which can suitably represent both attributes and relations among layout components. Two alternative page layout representations have been investigated: Purely symbolic and mixed symbolic-numeric. In order to handle the latter representation, the specialization operator of the first-order learning algorithm has been extended. The empirical results confirmed that the on-line local discretization of numerical attributes greatly simplifies the learned rules and generally decreases the learning time with no loss of predictive accuracy. As future work, the application of the first-order learning algorithm to different levels of the layout structure (e.g., frame1) will be investigated.

Appendix A

Proof of Theorem 1: Let S be an ordered list of N real values with positive or negative label. S can be partitioned into two intervals S_L and S_R. Without loss of generality, suppose that S_L is the admissible interval. We want to show that the minimizing threshold T that partitions S in S_L and S_R cannot occur within a group of adjacent cases, all of which have the same label. In general, assume that T occurs somewere within a sequence of n_j vaues with the same label, where $n_j \geq 2$. Without loss of generality, assume that this label is $+$. Assume that n^+ cases in this sequence of n_j positive cases are less than T, $0 \leq n^+ \leq n_j$. Figure A1 illustrates the situation. Our sequence consists of n_j examples that have values greater than T_1 and less than T_2, where T_1 and T_2 are boundary points.

Let L be cases in S with values $< T_1$, and R cases in S with values $> T_2$, where $0 \leq L, R \leq N - n_j$. Note that $n_j + L + R = N$ by definition. Let $L^+(L^-)$ be the number of positive (negative) cases in S with values less than T. We will show that

$$H(L^+ + n^+, L^-) = \frac{L^-}{L^+ + n^+} \mathit{info}(L^+ + n^+, L^-)$$

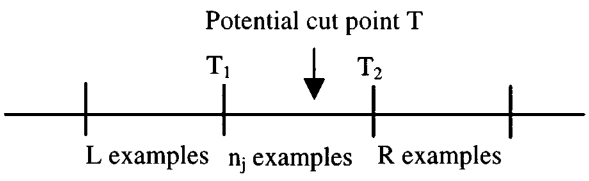


Figure A1. Modeling a possible partition.

is minimized at $n^+ = 0$ or at $n^+ = n_j$, thus forcing T to coincide with one of the boundary points T_1 or T_2.

1st case: $L^+ \geq L^-$
The addition of n^+ further positive cases to the initial set of L^+ positive cases causes the decrease of both the entropy $\mathit{info}(L^+ + n^+, L^-)$ and of the ratio

$$\frac{L^-}{L^+ + n^+}$$

Therefore, $H(L^+ + n^+, L^-)$ is strictly decreasing and the minimum is obtained for $n^+ = n_j$.

2nd case: $L^+ < L^-$
Let us compute the first derivative of $H(L^+ + n^+, L^-)$ with respect to n^+.

$$\frac{d}{dn^+} H(L^+ + n^+, L^-)$$
$$= -\frac{L^-}{(L^+ + n^+)^2}\left[-\log\frac{L^+ + n^+}{L + n^+} + \frac{(L^-)^2}{(L^+ + n^+)^2}\log\frac{L^-}{L + n^+}\right]$$

Therefore, we have

$$\frac{d}{dn^+} H(L^+ + n^+, L^-) = 0$$

only if

$$\log\frac{L^+ + n^+}{L + n^+} = \frac{(L^-)^2}{(L^+ + n^+)^2}\log\frac{L^-}{L + n^+}$$

This equality holds when $L^+ + n^+ = L^-$, that is $n^+ = L^- - L^+$. Note that this difference is not negative since $L^+ < L^-$.

When $n^+ < L^- - L^+$ then $L^+ + n^+ < L^-$, thus we have the following inequalities:

$$\log\frac{L^+ + n^+}{L + n^+} < \log\frac{L^-}{L + n^+} < \frac{(L^-)^2}{(L^+ + n^+)^2}\log\frac{L^-}{L + n^+}$$

since now $\frac{(L^-)^2}{(L^+ + n^+)^2} > 1$.

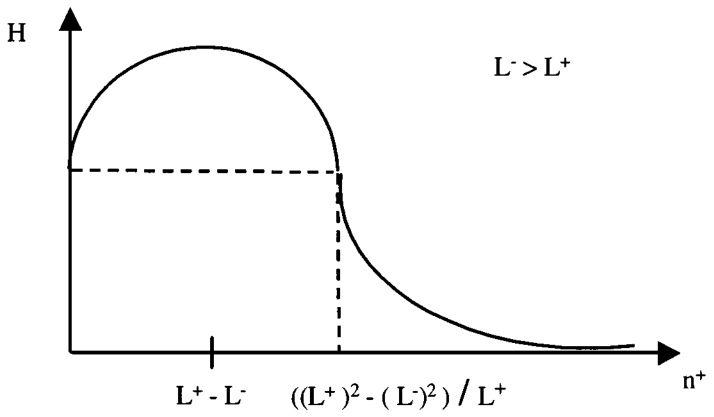


Figure A2. Behaviour of the function $H(L^+ + n^+, L^-)$ when $L^+ < L^-$.

Therefore, when $n^+ < L^- - L^+$ we have:

$$\frac{d}{dn^+} H(L^+ + n^+, L^-) > 0$$

that is $H(L^+ + n^+, L^-)$ is strictly increasing in the right open interval $[0, L^- - L^+[$.

Furthermore, when $n^+ > L^- - L^+$, that is $n^+ + L^+ > L^-$, the following inequalities hold:

$$\log \frac{L^+ + n^+}{L + n^+} > \log \frac{L^-}{L + n^+} > \frac{(L^-)^2}{(L^+ + n^+)^2} \log \frac{L^-}{L + n^+}$$

since now $\frac{(L^-)^2}{(L^+ + n^+)^2} < 1$.

Therefore, when $n^+ > L^- - L^+$ we have:

$$\frac{d}{dn^+} H(L^+ + n^+, L^-) < 0$$

that is $H(L^+ + n^+, L^-)$ is strictly decreasing in the open interval $]L^- - L^+, +\infty[$.

To sum up, the entropic function H increases up to a maximum in $L^- - L^+$, then decreases towards zero as n^+ increases (see figure A2).

It is easy to prove that $H(L^+, L^-) = H(L^+ + n^+, L^-)$ when

$$n^+ = \frac{(L^-)^2 - (L^+)^2}{L^+}$$

Therefore, we can conclude that if $T_2 < \frac{(L^-)^2 - (L^+)^2}{L^+}$ then $H(L^+ + n^+, L^-)$ is minimized for $n^+ = 0$, while if $T_2 > \frac{(L^-)^2 - (L^+)^2}{L^+}$ then it is minimized for $n^+ = n^j$. In the case $T_2 = \frac{(L^-)^2 - (L^+)^2}{L^+}$, the entropy is minimized in both extremes of the interval.

This proves that T must be a boundary point.

Acknowledgments

The authors would like to thank Francesco De Tommaso, Dario Gerbino, Ignazio Sardella, Giacomo Sidella, Rosa Maria Spadavecchia and Silvana Spagnoletta for their contribution to the development of WISDOM++. Thanks are also due to the authors of the systems C4.5 and ITI. We are grateful to Lynn Rudd for her help in re-reading the first draft of the paper.

Notes

1. The data set is available in the UCI Machine Learning Repository, www.ics.uci.edu/∼mlearn/MLRepository.html
2. A rule is *complete* if it covers all positive examples.
3. A rule is *consistent* if it covers no negative example.
4. Published by Springer-Verlag in the series Lecture Notes on Artificial Intelligence, Vol. 869.
5. Published in the period January–June 1996.

References

Altamura, O., Esposito, F., and Malerba, D. (1999). WISDOM++: An Interactive and Adaptive Document Analysis System. In *Proc. of the 5th Int. Conf. on Document Analysis and Recognition* (pp. 366–369). Los Alamitos: IEEE Computer Society Press.

Bayer, T., Bohnacher, U., and Mogg-Schneider, H. (1994). InforPortLab: An Experimental Document Analysis System. In *Proc. of the IAPR Workshop on Document Analysis Systems*, Kaiserslautern, Germany.

Bergadano, F. and Bisio, R. (1988). Constructive Learning with Continuous-Valued Attributes. In B. Bouchon, L. Saitta, and R.R. Yager (Eds.), *Lecture Notes in Computer Science, Vol. 313*: *Uncertainty and Intelligent Systems* (pp. 54–162). Berlin: Springer-Verlag.

Botta, M. and Giordana, A. (1991). Learning Quantitative Features in a Symbolic Environment. In Z.W. Ras and M. Zemankova (Eds.), *Lecture Notes in Artificial Intelligence, Vol. 542*: *Methodologies for Intelligent Systems* (pp. 296–305). Berlin: Springer-Verlag.

Connell, J.H. and Brady, M. (1987). Generating and Generalizing Models of Visual Objects, *Artificial Intelligence*, 31(2), 159–183.

Dengel, A. and Barth, G. (1989). ANASTASIL: A Hybrid Knowledge-Based System for Document Layout Analysis. In *Proc. of the 6th Int. Joint Conf. on Artificial Intelligence* (pp. 1249–1254).

De Raedt, L. (1992). *Inductive Theory Revision*. London: Academic Press.

Dzeroski, S. and Bratko, I. (1996). Applications of Inductive Logic Programming. In L. De Raedt (Ed.), *Advances in Inductive Logic Programming* (pp. 65–81). Amsterdam: IOS Press.

Esposito, F., Lanza, A., Malerba, D., and Semeraro, G. (1997). Machine Learning for Map Interpretation: An Intelligent Tool for Environmental Planning, *Applied Artificial Intelligence: An International Journal*, 11(10), 673–696.

Esposito, F., Malerba, D., Semeraro, G., Annese, E., and Scafuro, G. (1990). Empirical Learning Methods for Digitized Document Recognition: An Integrated Approach to Inductive Generalization. In *Proc. of the 6th IEEE Conf. on Artificial Intelligence Applications* (pp. 37–45). Los Alamitos: IEEE Computer Society Press.

Esposito, F., Malerba, D., and Semeraro, G. (1993). Incorporating Statistical Techniques into Empirical Symbolic Learning Systems. In D.J. Hand (Ed.), *Artificial Intelligence Frontiers in Statistics* (pp. 168–181). London: Chapman & Hall.

Esposito, F., Malerba, D., and Semeraro, G. (1994). Multistrategy Learning for Document Recognition, *Applied Artificial Intelligence*, 8(1), 33–84.

Esposito, F., Malerba, D., Semeraro, Fanizzi, N., and Ferilli, S. (1998). Adding Machine Learning and Knowledge Intensive Techniques to a Digital Library Service, *International Journal on Digital Libraries*, 2(1), 3–19.

Fayyad, U.M. and Irani, K.B. (1992). On the Handling of Continuous-Valued Attributes in Decision Tree Generation. *Machine Learning*, 8, 87–102.

Fayyad, U., Piatesky-Shapiro, G., and Smyth, P. (1996). From Data Mining to Knowledge Discovery: An Overview. In U. Fayyad, G. Piatesky-Shapiro, P. Smyth, and R. Uthurusamy (Eds.), *Advances in Knowledge Discovery and Data Mining* (pp. 1–35). AAAI/MIT Press.

Fisher, J.L., Hinds, S.C., and D'Amato, D.P. (1990). A Rule-Based System for Document Image Segmentation. In *Proc. of the 10th Int. Conf. on Pattern Recognition* (pp. 567–572). Los Alamitos: IEEE Computer Society Press.

Gemello, R., Mana, F., and Saitta, L. (1991). Rigel: An Inductive Learning System, *Machine Learning*, 6(1), 7–35.

Helft, N. (1987). Inductive Generalization: A Logical Framework. In I. Bratko and N. Lavrac (Eds.), *Progress in Machine Learning—Proc. of the EWSL87* (pp. 149–157). London: Sigma Press.

Lavrac, N. and Dzeroski, S. (1994). *Inductive Logic Programming: Techniques and Applications*. Chichester: Ellis Horwood.

Lavrac, N., Dzeroski, S., and Bratko, I. (1996). Handling Imperfect Data in Inductive Logic Programming. In L. De Raedt (Ed.), *Advances in Inductive Logic Programming* (pp. 48–64). Amsterdam: IOS Press.

Malerba, D., Semeraro, G., and Bellisari, E. (1995). LEX: A Knowledge-Based System for the Layout Analysis. In *Proc. of the 3rd Int. Conf. on the Practical Application of Prolog* (pp. 429–443).

Malerba, D., Esposito, F., Semeraro, G., and Caggese, S. (1997a). Handling Continuous Data in Top-Down Induction of First-Order Rules. In M. Lenzerini (Ed.), *Lecture Notes in Artificial Intelligence, Vol. 1321: AI*IA 97: Advances in Artificial Intelligence* (pp. 24–35). Berlin: Springer-Verlag.

Malerba, D., Esposito, F., Semeraro, G., and De Filippis, L. (1997b). Processing Paper Documents in WISDOM. In M. Lenzerini (Ed.), *Lecture Notes in Artificial Intelligence, Vol. 1321: AI*IA 97: Advances in Artificial Intelligence* (pp. 439–442). Berlin: Springer-Verlag.

Malerba, D., Semeraro, G., and Esposito, F. (1997c). A Multistrategy Approach to Learning Multiple Dependent Concepts. In C. Taylor and R. Nakhaeizadeh (Eds.), *Machine Learning and Statistics: The Interface* (pp. 87–106). London: Wiley.

Mitchell, T. (1997). *Machine Learning*. New York: McGraw Hill.

Nagy, G., Seth, S.C., and Stoddard, S.D. (1992). A Prototype Document Image Analysis System for Technical Journals, *IEEE Computer*, 25(7), 10–22.

Orkin, M. and Drogin, R. (1990). *Vital Statistics*. New York: McGraw Hill.

Pagurek, B., Dawes, N., Bourassa, G., Evans, G., and Smithers, P. (1990). Letter Pattern Recognition. In *Proc. of the 6th Conf. on Artificial Intelligence Applications* (pp. 312–319). Los Alamitos: IEEE Computer Society Press.

Quinlan, J.R. (1993). *C4.5: Programs for Machine Learning*. San Mateo: Morgan Kaufmann.

Quinlan, J.R. and Cameron-Jones, R.M. (1993). FOIL: A Midterm Report. In P.B. Brazdil (Ed.), *Lecture Notes in Artificial Intelligence, Vol. 667: Machine Learning: ECML-93* (pp. 3–20). Berlin: Springer-Verlag.

Tang, Y.Y., Yan, C.D., and Suen, C.Y. (1994). Document Processing for Automatic Knowledge Acquisition, *IEEE Trans. on Knowledge and Data Engineering*, 6(1), 3–21.

Utgoff, P.E. (1994). An Improved Algorithm for Incremental Induction of Decision Trees. In *Proc. of the 11th Int. Conf. on Machine Learning* (pp. 318–325). San Francisco: Morgan Kaufmann.

Wang, D. and Srihari, R.N. (1989). Classification of Newspaper Image Blocks Using Texture Analysis, *Computer Vision, Graphics, and Image Processing*, 47, 327–352.

Wong, K.Y., Casey, R.G., and Wahl, F.M. (1982). Document Analysis System, *IBM J. of Research Development*, 26(6), 647–656.