
Learning Logic Programs for Layout Analysis Correction

Margherita Berardi
Michelangelo Ceci
Floriana Esposito
Donato Malerba

BERARDI@DI.UNIBA.IT
CECI@DI.UNIBA.IT
ESPOSITO@DI.UNIBA.IT
MALERBA@DI.UNIBA.IT

Dipartimento di Informatica, University of Bari, Via Orabona 4, 70126 Bari, Italy

Abstract

Layout analysis is the process of extracting a hierarchical structure describing the layout of a page. In the system WISDOM++, the layout analysis is performed in two steps: firstly, the global analysis determines possible areas containing paragraphs, sections, columns, figures and tables, and secondly, the local analysis groups together blocks that possibly fall within the same area. The result of the local analysis process strongly depends on the quality of the results of the first step. We investigate the possibility of supporting the user during the correction of the results of the global analysis. This is done by automatically generating training examples of action selections from the sequence of user actions, and then by learning action selection rules for layout correction. Rules are expressed as a logic program whose induction demands the careful application of ILP techniques. Experimental results on a set of multi-page documents shed evidence on the difficulty of the learning task tackled and pose new problems in learning control rules for adaptive interfaces.

1. Background and motivation

Strategies for the extraction of layout analysis have been traditionally classified as top-down or bottom-up (Srihari & Zack, 1986). In top-down methods the document image is repeatedly decomposed into smaller and smaller components, while in bottom-up methods basic layout components are extracted from bitmaps and then grouped together into larger blocks on the basis of their characteristics. In WISDOM++, a document image analysis system that can transform paper documents into XML format (Altamura, Esposito & Malerba, 2001), the applied page decomposition method is hybrid, since it combines a top-down approach to segment the document

image and a bottom-up layout analysis method to assemble basic blocks into frames.

Some attempts to learn the layout structure from a set of training examples have also been reported in the literature (Dengel, 1993; Dengel & Dubiel, 1995; Kise, 1993; Walischewski 1997). They are based on ad-hoc learning algorithms, which learn particular data structures, such as geometric trees and tree grammars. Results are promising, although it has been proven that good layout structures could also be obtained by exploiting generic knowledge on typographic conventions (Esposito, Malerba & Semeraro, 1995). This is the case of WISDOM++, which analyzes the layout in two steps:

1. A *global analysis* of the document image, in order to determine possible areas containing paragraphs, sections, columns, figures and tables. This step is based on an iterative process, in which the vertical and horizontal histograms of text blocks are alternately analyzed, in order to detect columns and sections/paragraphs, respectively.
2. A *local analysis* of the document to group together blocks that possibly fall within the same area. Generic knowledge on west-style typesetting conventions is exploited to group blocks together, such as “the first line of a paragraph can be indented” and “in a justified text, the last line of a paragraph can be shorter than the previous one”.

Experimental results proved the effectiveness of this knowledge-based approach on images of the first page of papers published in conference proceedings and journals (Altamura, Esposito & Malerba, 2001). However, performance degenerates when the system is tested on intermediate pages of multi-page articles, where the structure is much more variable, due to the presence of formulae, images, and drawings that can stretch over more than one column, or are quite close. The majority of errors made by the layout analysis module were in the global analysis step, while the local analysis step performed satisfactorily when the result of the global analysis was correct.

In this paper, we investigate the possibility of supporting the user during the correction of the results of the global analysis. This is done by allowing the user to correct the results of the global analysis and then by learning rules for layout correction from his/her sequence of actions.

This approach differs from those that learn the layout structure from scratch, since the goal is to learn what the “errors” are which are performed by the automated global analysis process that the user corrects. Other document processing systems allow users to correct the result of the layout analysis; nevertheless WISDOM++ is the only one that tries to learn corrective actions from user interaction with the system.

The problem we intend to solve is also different from learning search control rules in Artificial Intelligence Planning (AIP). In AIP an initial state and a goal are given and the task is to find a sequence of actions that maps the state into the goal by searching through a list of domain actions. Control rules are learned from plans (automatically) generated by a given planner for a set of initial states and goals. They are then used either to substitute the planner (Khardon, 1999) or to assist the search (Huang, Selman & Kautz, 2000). In our case, plans are manually generated by the user when he/she corrects the result of the global layout analysis. The initial states are automatically determined by the system WISDOM++, while the goals are implicitly defined by the user during his/her manual correction. Goals are not known for testing documents, therefore, the result of the learning algorithm can neither replace the planner nor assist it. Inductive learning algorithms can only be applied to generalize the set of preconditions for the applicability of domain actions, independently of the final goal.

In the following section, we describe the layout correction operations. Section 3 explains the automated generation of training examples, while Section 4 briefly introduces the learning strategy. Experimental results are reported in Section 5 and discussed in Section 6, together with current limitations and new research goals.

2. Correcting the layout

Global analysis aims to determine the general layout structure of a page and operates on a tree-based representation of nested columns and sections. The levels of columns and sections are alternated (Figure 1), which means that a column contains sections, while a section contains columns. At the end of the global analysis, the user can only see the sections and columns that have been considered atomic, that is, not subject to further decomposition (Figure 2). The user can correct this result by means of three different operations:

- Horizontal splitting: a column/section is cut horizontally.
- Vertical splitting: a column/section is cut vertically.
- Grouping: two sections/columns are merged together.

The cut point in the two splitting operations is automatically determined by computing either the horizontal or the vertical histogram on the basic blocks returned by the segmentation algorithm. The horizontal (vertical) cut point corresponds to the largest gap between two consecutive bins in the horizontal (vertical) histogram. Therefore, splitting operations can be described by means of a unary function, $split(X)$, where X represents the column/section to be split and the range is the set $\{horizontal, vertical, no_split\}$.

The grouping operation, which can be described by means of a binary predicate $group(A,B)$, is applicable to two sections (columns) A and B and returns a new section (column) C , whose boundary is determined as follows. Let $(left_X, top_X)$ and $(bottom_X, right_X)$ be the coordinates of the top-left and bottom-right vertices of a column/section X , respectively.¹ Then:

$$left_C = \min(left_A, left_B), \quad right_C = \max(right_A, right_B), \\ top_C = \min(top_A, top_B), \quad bottom_C = \max(bottom_A, bottom_B).$$

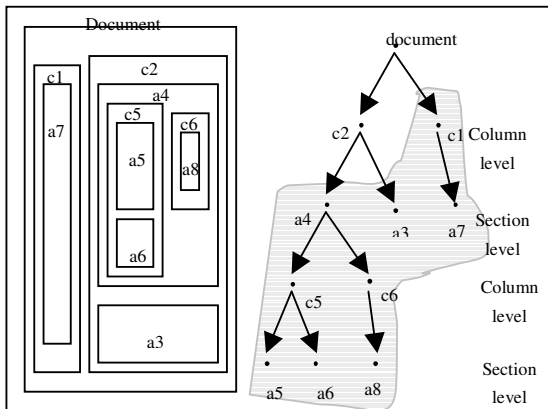


Figure 1. Layout tree. Columns and sections are alternated.

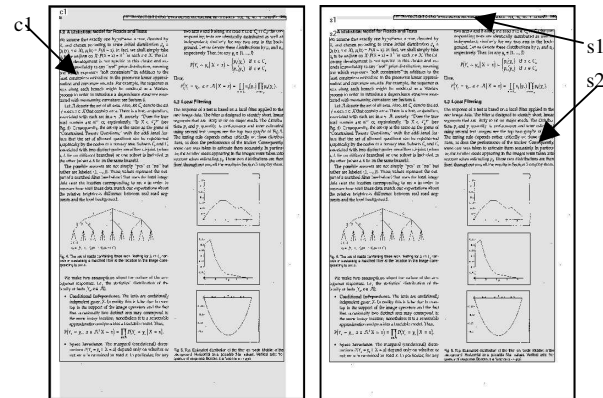


Figure 2. Results of the global analysis process: one column (left) includes two sections (right). The result of the local analysis process is reported in the background.

¹ The origin of the coordinate system is in the top left-hand corner.

Grouping is possible only if the following two conditions are satisfied:

1. C does not overlap another section (column) in the document.
2. A and B are nested in the same column (section).

After each splitting/grouping operation, WISDOM++ recomputes the result of the local analysis process, so that the user can immediately perceive the final effect of the requested correction and can decide whether to confirm the correction or not.

3. Representing corrections

From the user interaction, WISDOM++ implicitly generates some training observations describing when and how the user intended to correct the result of the global analysis. These training observations are used to learn correction rules of the result of the global analysis, as explained in the next section.

The simplest representation describes, for each training observation, the page layout at the i -th correction step and the corrective operation performed by the user on that layout. Therefore, if the user performs $n-1$ corrective operations, n observations are generated. The last one corresponds to the page layout accepted by the user. In the learning phase, this representation may lead the system to generate rules which only take into account the user's exact correction sequence. However, several alternative correction sequences, which lead to the same result, may also be possible. If they are not considered, the learning strategy will suffer from data overfitting problems. This issue was already discussed in a preliminary work (Malerba, Esposito & Altamura, 2002).

A more sophisticated representation, which takes into account alternative correction sequences, is based on the commutativity of some corrective operations. When the sequential order of two or more operations is irrelevant for the final result, only one training observation can be generated, such that all commutative operations are associated to the page layout. However, in this representation, it is crucial to use a method for the discovery of alternative correction sequences.

Before formally describing the method, some useful notations are introduced.

Let $\Lambda(P)$ be the space of possible layout trees of a page P . Each operation can be defined as a function $O: \Lambda(P) \rightarrow \Lambda(P)$, which transforms a layout tree $T_1 \in \Lambda(P)$ into the tree $T_2 = O(T_1) \in \Lambda(P)$, such that T_2 is derived from T_1 by applying a split/group operator to a specific column or section. Each function O can be partially defined, since not all operations are admissible (see previous section). The set of admissible operations for $T_1 \in \Lambda(P)$ will be denoted as $A(T_1)$.

Def. Independence between operations

Let O_1 and O_2 be two operations defined on $\Lambda(P)$ and $T \in \Lambda(P)$. O_1 and O_2 are *independent* w.r.t the tree T iff $O_1 \in A(T)$, $O_1 \in A(O_2(T))$, $O_2 \in A(T)$, $O_2 \in A(O_1(T))$ and $O_1(O_2(T)) = O_2(O_1(T))$.

When a user corrects the global analysis, he/she performs a sequence of operations, not all of which are relevant. For example, if a user performs a grouping on two blocks and then a splitting on the resulting one, the page layout might remain unchanged and the two operations would annul each other. More formally, the following definitions can be given:

Def. Alternative sequence of operations

Let S_1 be a sequence of operations $S_1 := (O_1, O_2 \dots O_{m-1})$, $S_2 := (O_{i_1}, O_{i_2} \dots O_{i_{n-1}})$ is an alternative sequence of operations w.r.t. to S_1 and to an initial layout tree $T_1 \in \Lambda(P)$ iff $S_2(T_1) = S_1(T_1)$, that is $O_{m-1}(\dots O_2(O_1(T_1))) = O_{i_{n-1}}(\dots O_{i_2}(O_{i_1}(T_1)))$, and $n \leq m$.

Def. Minimal alternative sequence of operations

Let $\{S_1, S_2, \dots, S_p\}$ be the finite set of all alternative sequences of operations w.r.t. a sequence of operations S and an initial layout tree $T_1 \in \Lambda(P)$. $S_{i_1}, S_{i_2}, \dots, S_{i_k}$, $1 \leq k \leq p$ are *minimal* alternative sequences of operations iff $d := |S_{i_1}| = |S_{i_2}| = \dots = |S_{i_k}|$ and $\forall i \in [1..p]: d \leq |S_i|$.

When a user corrects the global analysis, WISDOM++ extracts a minimal alternative sequence of operations $S := (O_1, O_2 \dots O_{n-1})$, such that:

$$T_1 \xrightarrow{O_1} T_2 \xrightarrow{O_2} \dots \xrightarrow{O_{n-1}} T_n,$$

where T_1 is the initial layout tree produced by the global layout analysis, while T_n is the final layout tree that the user considers to be corrected. Henceforth, the sequence of operations used for the generation of training observations will be considered minimal.

The operation O_i can be *commuted* with the operation O_j , $1 \leq j < i \leq n$, if $\forall k, j \leq k < i$, O_k and O_i are independent w.r.t. T_k . When two operations can be commuted, a permutation S' of S exists, such that when it is applied to T_1 it produces T_n . The permutation S' shares a prefix and a suffix with S . This can be graphically represented as follows:

$$\begin{array}{c} S \\ S' \end{array} T_1 \xrightarrow{\text{prefix}} \dots \xrightarrow{O_j} T_j \xrightarrow{O_{j+1}} \dots \xrightarrow{O_{i-1}} T_{i-1} \xrightarrow{\text{suffix}} \dots \xrightarrow{O_{n-1}} T_n.$$

In an extreme situation, the prefix is only T_1 , while the suffix is only T_n . Therefore, the set of all permutations S' of S , obtained by commuting independent operations, define a *state graph*, where there is only one node with null indegree, namely T_1 , and only one node with no null outdegree, that is, T_n . Nodes in the state graph are labelled with layout trees, while directed edges are labelled with admissible operations. Every path from the root to the leaf represents a sequence of operations that, applied to the starting layout tree, generates the same corrected layout tree. An internal node T_j ($1 \leq j \leq n-1$) has a number of

children m , depending on the number of independent operations $\{O_1, O_2, \dots, O_m\}$ of S w.r.t. T_j , such that $O_i, 1 \leq i \leq m$ has not been applied in the path from the root T_1 to T_j .

For instance, if the user performs three operations to obtain the correct layout tree T_4

$$T_1 \xrightarrow{O_1} T_2 \xrightarrow{O_2} T_3 \xrightarrow{O_3} T_4$$

and if O_2 and O_3 can be commuted, then the state graph reported below can be built:

$$T_1 \xrightarrow{O_1} T_2 \begin{array}{l} \xrightarrow{O_2} T_3 \xrightarrow{O_3} T_4 \\ \xrightarrow{O_3} T_5 \xrightarrow{O_2} T_4 \end{array}$$

The graph shows the sequences of operations allowed to obtain the layout tree T_4 .

Wisdom++ stores the sequence of operations performed by the user O_1, O_2, \dots, O_n and the layout trees T_1 and T_n . On the basis of this information it builds the state graph and computes the minimal alternative sequences of operations in order to generate a set of training observations. Minimality of the sequence of operation is necessary to prevent the system from generating “clashing” examples, that is, different operations applied to the same component of identical layout structures.

The definition of a suitable representation language for the global layout structure is a key issue. In this work we restrict this representation to the lowest column and section levels in the tree structure extracted by the global analysis and we deliberately ignore other levels and their composition hierarchy. Nevertheless, describing this portion of the layout structure is not straightforward, since the columns and sections are spatially related and the propositional representation languages cannot render these relations. Therefore, we resort to the application of a first-order logic language, where unary function symbols, called *attributes*, are used to describe properties of a single layout component (e.g., height), while binary predicate and function symbols, called *relations*, are used to express spatial relationships among layout components (e.g., part-of).

The following is an example of a training observation automatically generated by WISDOM++:

```
split(c1)=horizontal, group(s1,s2)=false,
split(s1)=no_split, split(s2)=no_split ←
width_s(s1)=552, width_s(s2)=552,
width_c(c1)=552, height_s(s1)=8,
height_s(s2)=723, height_c(c1)=852,
x_pos_centre_s(s1)=296, x_pos_centre_s(s2)=296,
x_pos_centre_c(c1)=296, y_pos_centre_s(s1)=22,
y_pos_centre_s(s2)=409, y_pos_centre_c(c1)=426,
s_on_top_s(s1,s2)=true, part_of(c1,s1)=true,
part_of(c1,s2)=true, no_blocks_s(s1)=2,
no_blocks_s(s2)=108, no_blocks_c(c1)=110,
per_text_s(s1)=100, per_text_s(s2)=83,
per_text_c(c1)=84.
```

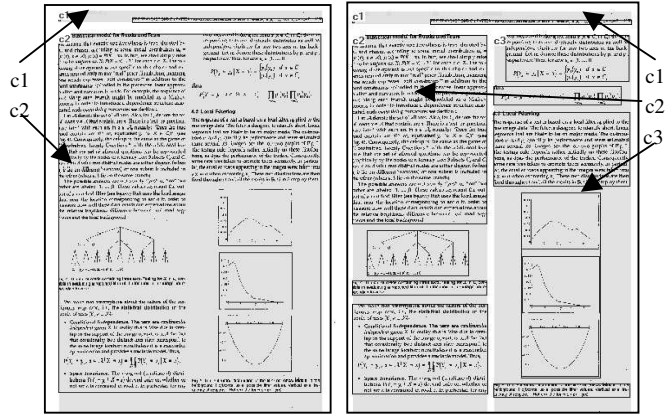


Figure 3. Horizontal split of the column (left) and vertical split of column c2 (right). The result of the layout analysis process is in the background.

This is a multiple-head ground clause, which has a conjunction of literals in the head. It describes the corrections applicable to the page layout in Figure 1, where two sections and one column were originally found. The horizontal splitting of the column ($split(c1)=horizontal$) is the first correction performed by the user (Figure 3). Since no other operation is independent of it, it is the only positive example of split/group operation. Indeed, the representation by multiple-head clauses allows us to easily express a form of concurrency in the execution of independent operations. The multiple-head clause also shows that the two sections $s1$ and $s2$ should be neither split (literals $split(s1)=no_split$ and $split(s2)=no_split$) nor grouped (literal $group(s1,s2)=false$). Many other literals, such as $group(c1,s2)=false$, $group(s1,c1)=false$ and $group(c1,c1)=false$, have not been generated, since they do not represent admissible operations.

The body represents the layout tree as attributes and relations. The prefixes (suffixes) $c_$ and $s_$ ($_c$ and $_s$) of function symbols specify whether the involved arguments are columns or sections. The column to be split is 552 pixels wide and 852 pixels high, has a center located at the point (296,426), and includes 110 basic blocks and the two sections $s1$ and $s2$, which are one on top of the other. The percentage of the area covered by the text blocks, enclosed by the column, is 84%.

4. The learning strategy

The inductive learning problem to be solved concerns the concepts $split(X)=horizontal$, $split(X)=vertical$ and $group(X,Y)=true$, since we are interested in finding rules which predict both when to split a column/section horizontally/vertically and when to group two columns/sections. No rule is generated for the case $split(X)=no_split$ and $group(X,Y)=false$. In other words, we are interested in action selection rules and not in action rejection rules.

Rules for the correction of the layout analysis can be automatically learned by means of an ILP or relational learning system. In this work, the system ATRE (<http://www.di.uniba.it/~malerba/software/atre>) has been used (Malerba, Esposito & Lisi, 1998). It solves the following learning problem:

Given

- a set of concepts C_1, C_2, \dots, C_r to be learned,
- a set of observations O described in a language L_O ,
- a background knowledge BK expressed in a language L_{BK} ,
- a language of hypotheses L_H ,
- a generalization model Γ over the space of hypotheses,
- a user's preference criterion PC ,

Find

a (possibly recursive) logical theory T for the concepts C_1, C_2, \dots, C_r , such that T is complete and consistent with respect to O and satisfies the preference criterion PC .

The *completeness* property holds when the theory T explains all observations in O of the r concepts C_i , while the *consistency* property holds when the theory T explains no counter-example in O of any concept C_i . The satisfaction of these properties guarantees the correctness of the induced theory with respect to O .

The preservation of the consistency property explains why it is important not to generate clashing examples, by computing minimal sequences of operations, as observed in the previous section.

In ATRE, observations are represented by means of ground multiple-head clauses, called *objects*. All literals in the head of the clause are called *examples* of the concepts C_1, C_2, \dots, C_r . They can be considered either positive or negative according to the learning goal. In this application domain, the set of concepts to be learned is $split(X)=horizontal, split(X)=vertical, group(X,Y)=true$ and no background knowledge is available.

The generalization model provides the basis for organizing the search space, since it establishes when a hypothesis explains a positive/negative example and when a hypothesis is more general/specific than another. The generalization model adopted by ATRE, called *generalized implication*, is explained in (Esposito, Malerba & Lisi, 2000).

The preference criterion PC is a set of conditions used to discard/favour some solutions while searching for a consistent hypothesis. In this work short rules, which explain a high number of positive examples and a low number of negative examples, are preferred.

It is noteworthy that the learning strategy implemented in ATRE is not affected by imbalanced data, since the goal is not to maximize the accuracy (Provost, 2000) but to generate consistent theories. As explained in the next section, this is an important aspect of our application. Other systems that suffer from this problem, such as Tilde

(Blokkeel & De Raedt, 1998), proved unsuitable, since they generated only a trivial classifier covering all positive and negative examples when tested on this learning task.

5. Experimental results

To investigate the applicability of the proposed solution we considered twenty-four papers, published as either regular or short, in the IEEE Transactions on Pattern Analysis and Machine Intelligence, in the January and February issues of 1996. Each paper is a multi-page document; therefore, we processed 210 document images in all. For 148 document images the user performed some corrections. The average number of corrections is 2.47 (i.e. 366/148) per corrected page. In fact, some intermediate pages of multi-page documents are the most critical and may require several operations to correct the column/section structure. The number of objects for ATRE corresponds to the total number of nodes in the state graph of all pages, namely 210 leaves (one for each corrected page layout) and 514 internal nodes. The total number of examples is 36,549, which corresponds to the total number of literals in the multiple-head clauses. Given the set of concepts to be learned, only 736 out of 36,549 examples are positive and correspond to either corrective actions actually performed by the user (vertical/horizontal splitting or grouping) or corrective actions generated automatically by the system, thanks to its independence property. Negative examples are implicitly created from what the user did not do.

The performance of the learning task is evaluated by means of a 6-fold cross-validation, that is, the set of twenty-four documents is first divided into six blocks (or folds) of four documents (Table 1), and then, for every block, ATRE is trained on the remaining blocks and tested on the hold-out block. For each learning problem, the number of omission/commission errors is recorded. *Omission* errors occur when corrective actions on the page layout are missed, while *commission* errors occur when wrong actions are "recommended" by a rule.

Experimental results are reported in Table 2 for each trial, and the average number of omission and commission errors is also given. Two conclusions can be drawn from Table 2. Firstly, there is a high level of variability among the trials. For instance, the percentage of omission errors of the rule for grouping in the second trial is relatively low (about 10.8%), while the same percentage for the third trial is quite high (about 73.5%). A possible explanation might be the heterogeneous correction procedures adopted by the different users who worked on the correction of the document layouts. Secondly, the percentage of commission errors is very low with respect to the percentage of omission errors. This means that learned rules are generally specific, because of the low percentage of positive examples (about 2%), with respect to the total number of training examples.

Table 1. Distribution of pages and examples per document grouped by six folds.

| | Name of the multi-page document | No. of pages | No. of literals | No. of horizontal splits | No. of vertical splits | No. of groups | Total number of examples |
|------------|---------------------------------|--------------|-----------------|--------------------------|------------------------|---------------|--------------------------|
| Fold No. 1 | TPAMI4 | 14 | 3534 | 7 | 6 | 0 | 689 |
| | TPAMI10 | 3 | 1960 | 0 | 4 | 0 | 393 |
| | TPAMI12 | 6 | 4710 | 6 | 3 | 3 | 979 |
| | TPAMI24 | 6 | 6003 | 4 | 2 | 6 | 1248 |
| | Total | 29 | 16207 | 17 | 15 | 9 | 3309 |
| Fold No. 2 | TPAMI7 | 6 | 4790 | 4 | 1 | 16 | 855 |
| | TPAMI13 | 3 | 1601 | 0 | 0 | 2 | 309 |
| | TPAMI18 | 6 | 9638 | 2 | 13 | 10 | 1930 |
| | TPAMI23 | 7 | 4884 | 1 | 1 | 9 | 897 |
| | Total | 22 | 20913 | 7 | 15 | 37 | 3991 |
| Fold No. 3 | TPAMI2 | 8 | 8053 | 6 | 12 | 15 | 1453 |
| | TPAMI8 | 5 | 6242 | 5 | 13 | 4 | 1261 |
| | TPAMI11 | 6 | 1480 | 1 | 1 | 1 | 271 |
| | TPAMI14 | 10 | 9163 | 5 | 4 | 14 | 1666 |
| | Total | 29 | 24938 | 17 | 30 | 34 | 4651 |
| Fold No. 4 | TPAMI3 | 15 | 4626 | 10 | 7 | 5 | 831 |
| | TPAMI16 | 9 | 9814 | 2 | 14 | 50 | 1762 |
| | TPAMI19 | 20 | 24638 | 3 | 18 | 45 | 4641 |
| | TPAMI20 | 14 | 22556 | 2 | 20 | 42 | 4219 |
| | Total | 58 | 61634 | 17 | 59 | 142 | 11453 |
| Fold No. 5 | TPAMI1 | 14 | 13053 | 16 | 8 | 28 | 2473 |
| | TPAMI6 | 3 | 1475 | 12 | 7 | 0 | 223 |
| | TPAMI17 | 13 | 11710 | 81 | 14 | 0 | 1931 |
| | TPAMI22 | 5 | 3242 | 8 | 1 | 3 | 589 |
| | Total | 35 | 29480 | 117 | 30 | 31 | 5216 |
| Fold No. 6 | TPAMI5 | 6 | 5604 | 18 | 2 | 5 | 979 |
| | TPAMI9 | 5 | 2220 | 1 | 3 | 3 | 403 |
| | TPAMI15 | 15 | 31369 | 8 | 45 | 63 | 5642 |
| | TPAMI21 | 11 | 4680 | 4 | 1 | 5 | 905 |
| | Total | 37 | 43873 | 31 | 51 | 76 | 7929 |
| Total | 24 docs | 210 | 197045 | 206 | 200 | 329 | 36549 |

Table 2. Experimental results.

| Trial | Vertical splitting | | Horizontal splitting | | Grouping | |
|----------|--------------------|----------|----------------------|----------|----------|----------|
| | omiss. | comm. | omiss. | comm. | omiss. | comm. |
| 1 | 8/15 | 7/3294 | 10/17 | 11/3292 | 4/9 | 8/3300 |
| 2 | 11/15 | 23/3976 | 4/7 | 7/3984 | 4/37 | 21/3954 |
| 3 | 22/30 | 6/4621 | 12/17 | 4/4634 | 20/34 | 55/4617 |
| 4 | 33/59 | 32/11394 | 6/17 | 66/11436 | 94/142 | 89/11311 |
| 5 | 17/30 | 24/5186 | 96/117 | 33/5099 | 14/31 | 23/5185 |
| 6 | 28/51 | 9/7878 | 26/31 | 2/7898 | 18/77 | 77/7852 |
| Average% | 61.25 | 0.30 | 64.63 | 0.31 | 41.47 | 0.70 |
| St.dev.% | 9.43 | 0.19 | 18.22 | 0.26 | 20.99 | 0.36 |

Table 3. Experimental results. Complexity of the learned theory.

| Trial | Vertical Splitting | | Horizontal Splitting | | Grouping | |
|--------------------------------|--------------------|-------------------|----------------------|-------------------|---------------|-------------------|
| | No of clauses | Positive examples | No of clauses | Positive examples | No of clauses | Positive examples |
| 1 | 25 | 185 | 29 | 189 | 40 | 320 |
| 2 | 29 | 185 | 34 | 199 | 31 | 292 |
| 3 | 28 | 170 | 26 | 189 | 37 | 295 |
| 4 | 26 | 141 | 31 | 189 | 26 | 187 |
| 5 | 26 | 170 | 22 | 89 | 46 | 298 |
| 6 | 23 | 149 | 26 | 175 | 33 | 253 |
| Avg. No of clauses per example | 6.38 | | 6.09 | | 7.79 | |

Some statistics concerning the learned theories are reported in Table3. It is noteworthy that the average number of examples covered by a rule is between 6 and 8 for all three concepts. However, the variance is high. Some rules actually cover about 20-30% of training examples, while other cover only one example. Specificity of learned clauses is due to two factors: firstly, the limited number of positive examples used in the training set, and secondly, the fact that ATRE is asked to generate a *complete* theory, that is a set of clauses that explain *all* positive examples.

For the sake of completeness, some clauses for the three concepts are reported below:

- ```
split(X1)=vertical ←
 width_s(X1) ∈ [289.0..535.0],
 s_on_top_s(X2,X1)=true,
 per_text_s(X2) ∈ [75.0..88.0].
```
- ```
split(X1)=horizontal ←
    height_c(X1) ∈ [872.0..876.0],
    width_c(X1) ∈ [536.0..546.0],
    x_pos_centre_c(X1) ∈ [275.0..314.0].
```
- ```
group(X2,X1)=true ←
 c_to_right_c(X2,X1)=true,
 width_c(X2) ∈ [1.0..11.0],
 height_c(X2) ∈ [40.0..75.0].
```

The interpretation of these clauses is straightforward. For instance, the first clause states that «sections with width between 289 and 535 pixels that are under another section with a percentage of text between 75% and 88% should be vertically split». This rule captures the fact that a section spans over two columns (it is quite large, indeed) in documents that are organized in two columns. The geometrical relation with another section whose content is mostly text excludes the situation in which figures spanning over two columns are also present in the training documents. The second clause states that high and large columns, centered with respect to the page, should be horizontally split to separate the running head from the body of the paper. It is noteworthy that the first and third clauses involve some relations and could be generated only by relational learning systems such as ATRE.

## 6. Discussion

Experimental results prove the difficulty of this learning task, which is characterized by a relatively low percentage of positive training examples for sometimes-complex correction tasks. Learned theories also show that for this task it is important to consider the spatial relations existing between columns and sections. Such spatial relations can be properly modelled in a first-order logic formalism, which requires the application of ILP or relational learning systems. However, the training set is very imbalanced and this prevents the applicability of some learning algorithms that optimize the predictive accuracy, since they return trivial “no-action” rules, which are actually more than 98% correct.

A further difficulty lies in the complexity of the descriptions generated for each multi-page document. This poses efficiency problems for the learning system, which must be able to handle both numeric and symbolic relational data.

Low performances may also be due to our attempt to learn pre-conditions of corrective actions, which are actually expressed as classification rules. However, corrective actions modify the state of the world, that is, the layout structure, while no state change is modelled by classification rules. Predicting the result of an action (how the state of the world changes) might be equally important.<sup>2</sup> However, this means that we have to learn both pre- and post-conditions of a corrective action.

Our problem shares some similarities with that faced by Lorenzo and Otero (2001), who also investigated issues related to learning action selection rules in Situation Calculus. More precisely, their problem was to learn the predicates *select*(*a*, *p*, *s*) and *reject*(*a*, *p*, *s*), where *p* is a plan and *a* is an action (in our case, horizontal/vertical split or group), executed in any situation *s* ∈ *p*. The background knowledge includes *holds/2* ground facts for fluents at every initial situation and every goal situation, the action theory of the domain in the form of a Situation Calculus program together with two universal frame axioms which describes how the world stays the same (they are necessary to solve the well-known “frame problem”). The main difference with our study is that the goal is unknown when WISDOM++ applies rules to automatically correct the layout tree. Therefore, no plan *p* can be considered to define the action selection rules, and no planning algorithm can be used to generate a plan of action. Other two methodological differences are:

- Reject rules, that is, conditions under which an operation must not be performed, are not learned.

---

<sup>2</sup> This seems to be especially important for the splitting operations, whose result cannot be exactly determined by considering only at the layout tree. Indeed, WISDOM++ determines the cut point of a horizontal/vertical split by computing the horizontal/vertical histogram on the basic blocks returned by the segmentation algorithm. This information used to determine cut points is external to the layout tree.

- The objects of the world (in our case columns and sections) should dynamically change, since they should be added or removed as the result of the execution of an action. For example, if the user horizontally splits a layout tree block  $T_1$ , the resulting layout tree  $O(T_1)$  has two new blocks, instead of the split one.

Currently, learned rules are operatively used in a production system with a forward chaining control structure. It is implemented with a theorem prover, using resolution to do forward chaining over a full first-order knowledge base. The system maintains a knowledge base (the working memory) of ground literals describing the layout tree. Ground literals are automatically generated by WISDOM++ after the execution of an operation. In each cycle, the system computes the subset of rules, whose condition part is satisfied by the current contents of the working memory (*match phase*). Conflicts are solved by selecting the first rule in the subset. Alternative strategies have not been tried, such as recency (prefer rules that refer to recently created working memory elements), or operation priority (which prefers splitting to grouping or viceversa). This is a limitation of our work and it underlines the more general problem of how to evaluate performance of the learned logical theory. Another aspect not investigated in this work concerns the non-termination of the match-act cycle of the production system. We never observed this problem in our experiments, but it is theoretically possible that two subsequent operations annul each other, thus causing non termination. A simple way to overcome this problem is to associate a step to each operation. In this case, the splitting operations are represented by a binary function, *split*(*X*,*S*), where *X* represents the column/section to be split and *S* is an ordinal number representing the correction process step. Clauses learned by ATRE are range-restricted, therefore the body of the clause will be forced to contain a literal of the form

$$step(S) \in [a..b],$$

which specifies the interval of correction steps in which the action can be performed. Every time WISDOM++ updates the working memory, a new ground literal *step*(*s*)=*k* should be added to. Finiteness of the intervals prevents the production system from having non-termination problems.

As a future work, we intend to investigate the possibility of formulating the problem as *goal-based*. Indeed, the performance of the complete layout analysis process (both global and local) can be evaluated on the basis of the number of layout components that can be associated with a logical label, which is an interpretation of its content (document understanding) (Malerba et al., 2001). This means that the better the result of the document understanding process, the better the result of the layout analysis process. Assuming that the user corrected the result of the global analysis to separate/group layout components with a different/the same logical label, we

can define our abstract goal as maximizing the number of “splittable” layout components with univocally determined logical labels and minimizing the number of “groupable” layout components with the same label. In this case, a planner rather than a production system is required in the recognition phase. It would interface both the layout analysis module and the document understand module, which both use logic theories induced from a set of training examples.

Another open problem is related to dealing with different document classes, each of which has a different page layout structure. In this case, different classes may require different layout corrections. Therefore, the document class should be part of the preconditions of corrective actions. However, the correct document class can be typically recognized only when the layout structure is extracted. The formulation of the problem as goal-based can help, but the planner should now interface three modules (layout analysis, document classification and document understanding), thus making the planning problem even more difficult to solve and the task of learning search control rules much harder than learning those reported in the machine learning literature.

Finally, more extensive experiments will be performed on a set of documents made available by three European film archives participating to the project Collate.

## Acknowledgements

This work partially fulfills the research objectives set by the IST-1999-20882 project COLLATE (Collaboratory for Automation, Indexing and Retrieval of Digitized Historical Archive Material) funded by the European Union (<http://www.collate.de>). The authors also wish to thank Lynn Rudd for her help in reading the manuscript.

## References

- Altamura, O., Esposito, F., & Malerba, D. (2001). Transforming paper documents into XML format with WISDOM++. *International Journal on Document Analysis and Recognition*, 4, 2-17.
- Blockeel, H., & De Raedt, L. (1998). Top-down induction of first-order logical decision trees. *Artificial Intelligence*, 101, 285-297.
- Dengel, A. (1993). Initial learning of document structures. *Proceedings of the Second International Conference on Document Analysis and Recognition*, (pp. 86-90), Los Vaqueros, CA: IEEE Computer Society Press.
- Dengel, A., & Dubiel, F. (1995). Clustering and classification of document structure – A machine learning approach. *Proceedings of the Third International Conference on Document Analysis and Recognition*, (pp. 587-591), Los Vaqueros, CA: IEEE Computer Society Press.
- Esposito, F., Malerba, D., & Semeraro, G. (1995). A Knowledge-Based Approach to the Layout Analysis. *Proceedings of the Third International Conference on Document Analysis and Recognition*, (pp. 466-471), Los Vaqueros, CA: IEEE Computer Society Press.
- Esposito, F., Malerba, D., & Lisi, F.A. (2000). Induction of recursive theories in the normal ILP setting: issues and solutions. In J. Cussens & A. Frisch (Eds.), *Inductive Logic Programming*, Lecture Notes in Artificial Intelligence, 1866, (pp. 93-111), Berlin: Springer.
- Huang, Y.-C., Selman, B. & Kautz, H. (2000). Learning declarative control rules for constraint-based planning. *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, (pp. 825-830), San Francisco: Morgan Kaufmann.
- Khardon, R. (1999). Learning action strategies for planning domains. *Artificial Intelligence*, 113, 125-148.
- Kise, K. (1993). Incremental acquisition of knowledge about layout structures from examples of documents. *Proceedings of the Second International Conference on Document Analysis and Recognition*, (pp. 668-671), Los Vaqueros, CA: IEEE Computer Society Press.
- Lorenzo, D., & Otero, R. (2001) Learning Logic Programs for Action Selection in Planning. *Proceedings of the Third International Workshop on Extraction of Knowledge from Databases (EKDB'01)*, EPIA 2001, 10th Portuguese Conference on Artificial Intelligence.
- Malerba, D., Esposito, F., & Lisi, F.A. (1998). Learning recursive theories with ATRE. *Proceedings of the Thirteenth European Conference on Artificial Intelligence*, (pp. 435-439), John Wiley & Sons.
- Malerba, D., Esposito, F., Lisi, F.A., & Altamura, O. (2001). Automated Discovery of Dependencies Between Logical Components in Document Image Understanding. *Proceedings of the Sixth International Conference on Document Analysis and Recognition*, (pp. 174-178), Los Vaqueros, CA: IEEE Computer Society Press.
- Malerba, D., Esposito, F., & Altamura, O. (2002). Adaptive Layout Analysis of Document Images. In H.-S. Hacid, Z.W. Ras, D.A. Zighed, Y. Kodratoff, *Foundations of Intelligent Systems, 13th International Symposium, ISMIS'2002*, Lecture Notes in Artificial Intelligence, 2366, (pp. 526-534), Berlin: Springer.
- Provost, F. (2000). Learning with Imbalanced Data Sets. Invited paper for the *AAAI'2000 Workshop on Imbalanced Data Sets*.
- Srihari, S.N., & Zack, G.W. (1986). Document Image Analysis. *Proceedings of the Eighth International Conference on Pattern Recognition*, (pp. 434-436).
- Walischewski, H. (1997) Automatic knowledge acquisition for spatial document interpretation. *Proceedings of the Fourth International Conference on Document Analysis and Recognition*, (pp. 243-247), Los Vaqueros, CA: IEEE Computer Society Press.