Correcting the Document Layout: A Machine Learning Approach

Donato Malerba Floriana Esposito Oronzo Altamura
Michelangelo Ceci Margherita Berardi
Dipartimento di Informatica – Università degli Studi di Bari
via Orabona 4 - 70126 Bari
{malerba, esposito, altamura, ceci, berardi}@di.uniba.it

Abstract

In this paper, a machine learning approach to support the user during the correction of the layout analysis is proposed. Layout analysis is the process of extracting a hierarchical structure describing the layout of a page. In our approach, 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 allowing the user to correct the results of the global analysis and then by learning rules for layout correction from the sequence of user actions. Experimental results on a set of multi-page documents are reported and commented.

1. Background and motivation

Strategies for the extraction of layout analysis have been traditionally classified as top-down or bottom-up [10]. 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 [1], 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 [2,3,4,7,11]. 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 [5]. This is the case of WISDOM++, which analyzes the layout in two steps:

1. A global analysis, 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 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 [1]. 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 is different from those that learn the layout structure from scratch, since we try to correct the result of a global analysis returned by a bottom-up algorithm. Furthermore, we intend to capture knowledge on correcting actions performed by the user of the document image processing system. Other document processing systems allow users to correct the result of the layout analysis; nevertheless WISDOM++ is the only one that tries to learn correcting actions from user interaction with the system.

Proceedings of the Seventh International Conference on Document Analysis and Recognition (ICDAR 2003)
0-7695-1960-1/03 $17.00 © 2003 IEEE

IEEE COMPUTER SOCIETY

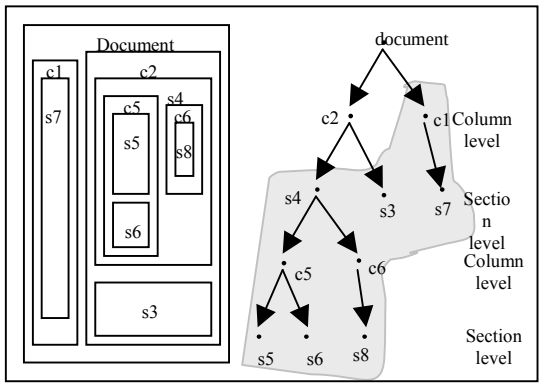

Fig. 1. Layout tree. Columns and sections are alternated.

In the following section, we describe the layout correction operations. The automated generation of training examples is explained in Section 3. Section 4 introduces the learning strategy, while Section 5 presents some experimental results.

2. Correcting the layout

Global analysis aims at determining 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. Then:

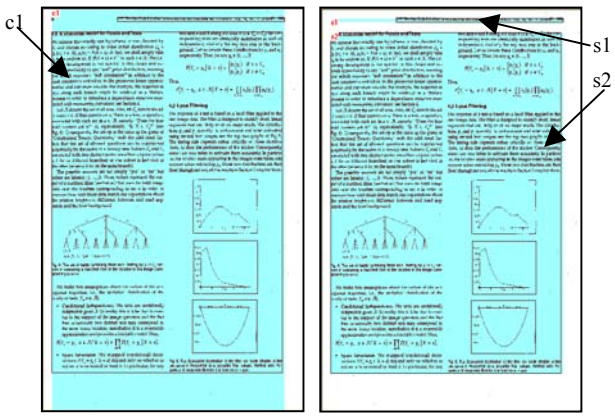

Fig. 2. Results of the global analysis process: one column (*left*) includes two sections (*right*). The result of the local analysis process (i.e., the frames) is reported in the background.

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

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 correcting operation performed by the user on that layout. Therefore, if the user performs *n-1* correcting 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 strictly take into account the exact user correction sequence. However, several alternative correction sequences, which lead to the same result, may be also possible. If they are not considered, the learning strategy will suffer from data overfitting problems. This issue was already discussed in a preliminary work [9].

A more sophisticated representation, which takes into account alternative correction sequences, is based on the

IEEE COMPUTER SOCIETY

commutativity of some correcting operations. When the sequential order of two ore 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/grouping 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 $S:=(O_1,O_2\ldots O_{n-1})$, such that:

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

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. The operation O_i can be *commuted* with the operation O_j, $1\le j<i\le n$, if $\forall k$, $j\le k<i$, 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:

$$\begin{array}{l} S \\ S' \end{array} \; T_1 \xrightarrow[\text{prefix}]{O_1 \;\ldots\; O_j} \begin{array}{l} \xrightarrow{O_{j+1}} \ldots \\ \xrightarrow{O_{kj+1}} \ldots \end{array} T_i \xrightarrow[\text{suffix}]{O_{i+1} \;\ldots\; O_{n-1}} T_n .$$

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 operation that, applied to the starting layout tree, generates the same layout tree. An internal node T_j ($1\le j\le n-1$) has a number of children m, depending on the number of independent operations $\{O_{l_1},O_{l_2},\ldots,O_{l_m}\}$ of S w.r.t. T_j, such that O_{l_i} $1\le i\le m$ has not been applied in the path from the root T_1 to T_j.

For instance, if the user performs 3 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:

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

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 $O_1,O_2\ldots O_{n-1}$and the layout trees T_1 and T_n. On the basis of this information it builds the state graph in order to generate a set of training observations.

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 feature-vector representation, typically adopted in statistical approaches, 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.
```

This is a multiple-head ground clause, which has a conjunction of literals in the head. It describes the correction 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). No other operations performed by the user can be applied as the first step, therefore, 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

IEEE COMPUTER SOCIETY

group(s1,s2)=false). Many other literals, such as *group(c1,s2)=false, group(s1,c1)=false* and

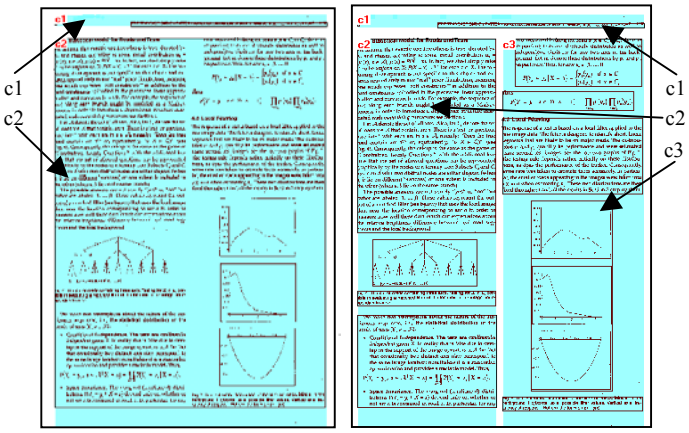

Fig. 3. Horizontal split of the column (*left*) and vertical split of column c2 (*right*). The result of the layout analysis process is in the background.

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 arguments involved 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%.

3. 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*.

Rules for the automated correction of the layout analysis can be automatically learned by means of a first-order learning system. The learning system ATRE has been used [8]. It solves the following learning problem:

Given

- a set of concepts C_1, C_2, …, 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, …, 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.

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, …, 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 are *split(X)=horizontal*, *split(X)=vertical*, *group(X,Y)=true*, since we are interested in finding rules which predict when to split horizontally/vertically or when to group two columns/sections. Therefore, no rule is generated for the case *split(X)=no_split* and *group(X,Y)=false*. Moreover, 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 [6].

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

4. 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 1996 issues. Each paper is a multi-page document; therefore, we processed 216 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 216 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 which correspond to

IEEE COMPUTER SOCIETY

either correcting actions actually performed by the user (vertical/horizontal splitting or grouping) or correcting actions generated automatically by the system, thanks to its independence.

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, 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 correct actions on page layout are missed, while *commission* errors occur when wrong actions are "recommended" by a rule.

Experimental results are reported in Table 1 for each trial, and the average number of omission and commission errors is also computed. Two conclusions can be drawn from Table 1. 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 different correction procedures adopted by the four 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. Experimental results.

Rule / Trial	*Vertical split*		*Horizontal split*		*Grouping*	
	omiss.	*Comm.*	*omiss.*	*comm.*	*omiss.*	*comm.*
1	8/15	1/3794	10/17	11/3292	4/9	8/3800
2	11/15	23/3976	4/7	7/3984	4/37	21/3954
3	22/30	6/4621	12/17	4/4634	25/34	9/4617
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	0.27	64.63	0.31	43.92	0.52
St.dev.%	9.42	0.21	18.21	0.25	24.05	0.31

5 Conclusions

This work presents an application of machine learning techniques to the problem of correcting the result of the global layout analysis process in WISDOM++. The proposed approach is an alternative to inducing the complete layout structure from a set of training examples. Training examples are automatically generated from the sequence of correcting operations performed by the user. The independence of the operations has also been considered to improve the representation of the corrections. Experimental results prove the difficulty of the learning task, which is characterized by a relatively low percentage of positive training examples for sometimes-complex correction tasks. This can be attributed both to the low average number of corrections performed by the user on each page and to the limited set of document images used in the experiments. A more extensive experimentation is planned for the next future.

Acknowledgements

This work fulfills the research objectives set by the IST-1999-20882 project COLLATE (Collaboratory for Annotation, Indexing and Retrieval of Digitized Historical Archive Material) funded by the European Union.

References

1. Altamura O., Esposito F., & Malerba D.: Transforming paper documents into XML format with WISDOM++, *Int. Journal on Document Analysis and Recognition*, 4(1), 2-17, 2001.
2. Akindele O.T., & Belaïd A.: Construction of generic models of document structures using inference of tree grammars, *Proc. of the 3rd Int. Conf. on Document Analysis and Recognition*, IEEE Computer Society Press, 206-209, 1995.
3. Dengel A.: Initial learning of document structures, *Proc. of the 2nd Int. Conf. on Document Analysis and Recognition*, IEEE Computer Society Press, 86-90, 1993.
4. Dengel A., & Dubiel F.: Clustering and classification of document structure – A machine learning approach, *Proc. of the 3rd Int. Conf. on Document Analysis and Recognition*, IEEE Computer Society Press, 587-591,1995.
5. Esposito F., Malerba D., & Semeraro G.: A Knowledge-Based Approach to the Layout Analysis, *Proc. of the 3rd Int. Conf. on Document Analysis and Recognition*, IEEE Computer Society Press, 466- 471, 1995.
6. Esposito F., Malerba D., & Lisi F.A.: Induction of recursive theories in the normal ILP setting: issues and solutions, in J. Cussens and A. Frisch (Eds.), *Inductive Logic Programming*, Lecture Notes in Artificial Intelligence, 1866, 93-111, Springer: Berlin, 2000.
7. Kise K.: Incremental acquisition of knowledge about layout structures from examples of documents. *Proc. of the 2nd Int. Conf. on Document Analysis and Recognition*, IEEE Computer Society Press, pp. 668-671, 1993.
8. Malerba D., Esposito F., & Lisi F.A.: Learning recursive theories with ATRE, *Proc. of the 13th European Conf. on Artificial Intelligence*, John Wiley & Sons, 435-439, 1998.
9. Malerba D., Esposito F., Altamura O.: Learning Rules for Layout Analysis Correction, *Proc. of the Int. Workshop on*

IEEE COMPUTER SOCIETY

Document Layout Interpretation and its Applications (DLIA'01), Seattle (WA), 2001.
10. Srihari S.N., & Zack G.W.: Document Image Analysis. *Proc. of the 8th Int. Conf. on Pattern Recognition*, 434-436, 1986.
11. Walischewski H.: Automatic knowledge acquisition for spatial document interpretation. *Proc. of the 4th Int. Conf. on Document Analysis and Recognition*, IEEE Computer Society Press, 243-247, 1997.

IEEE COMPUTER SOCIETY