Adaptive Layout Analysis of Document Images

Donato Malerba Floriana Esposito Oronzo Altamura

Dipartimento di Informatica, Università degli Studi di Bari,
via Orabona 4, I-70126 Bari – Italy
{malerba, esposito, altamura}@di.uniba.it

Abstract. Layout analysis is the process of extracting a hierarchical structure describing the layout of a page. In the document processing 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. 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 the sequence of user actions. Experimental results on a set of multi-page documents are reported.

1 Background and motivations

Processing document images, that is bitmaps of scanned paper documents, is a complex task involving many activities, such as preprocessing, segmentation, layout analysis, classification, understanding and text extraction [6]. Those activities are all important, although, the extraction of the right layout structure is deemed the most critical. *Layout analysis* is the perceptual organization process that aims at detecting structures among blocks extracted by the segmentation algorithm. The result is a hierarchy of abstract representations of the document image, called the *layout structure* of the document. 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 rectangular areas corresponding to groups of blocks.

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++ (www.di.uniba.it/~malerba/wisdom++/), a document image analysis system that can transform paper documents into either HTML or 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 of learning the layout structure from a set of training examples have also been reported in the literature [2,3,4,8,11]. They are based on ad-hoc learning algorithms, which learns 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* 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 either conference proceedings or 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 main source of the errors made by the layout analysis module was 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 means of two new system facilities:

1. the user can correct the results of the layout analysis by either grouping or splitting columns/sections, automatically produced by the global analysis;
2. the user can ask the system to learn grouping/splitting rules from his/her sequence of actions correcting the results of the layout analysis.

The proposed 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.

In the following section, a description of the layout correction operations is reported, and the automated generation of training examples is explained. Section 3 briefly introduces the learning system used to generate layout correction rules and presents some preliminary experimental results.

2 Correcting the results of the global analysis

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, 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 1). 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 binary function, namely, *split(X,S)*, where *X* represents the column/section to be split, *S* is an ordinal number representing the step of the correction process and the range of the split function is the set {*horizontal*, *vertical*, *no_split*}.

The grouping operation, which can be described by means of a ternary predicate *group(A,B,S)*, 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

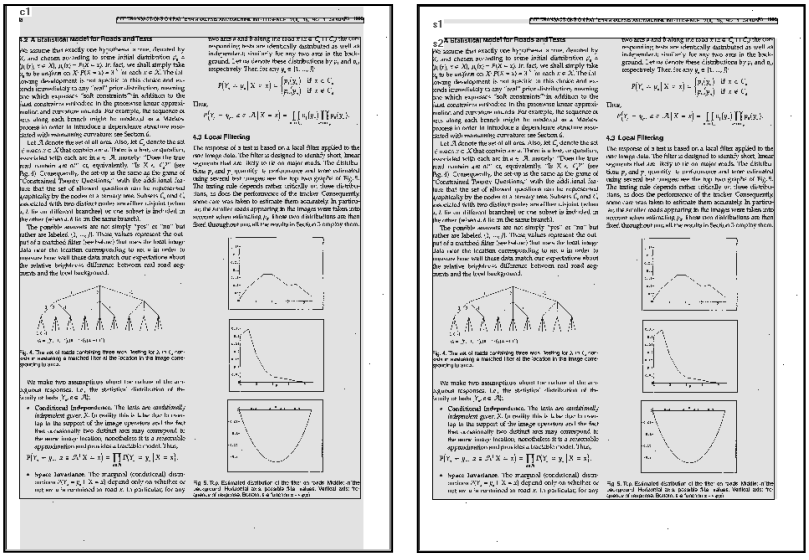

Fig. 1. Results of the global analysis process: one column (*left*) includes two sections (*right*). The result of the local analysis process (i.e., the frames) is in reported the background.

column/section *X*, respectively.[1] Then:

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

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 corrections and can decide whether to confirm the correction or not.

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 below.

3 Learning rules for layout correction

The inductive learning problem to be solved concerns the concepts *split(X,S)=horizontal*, *split(X,S)=vertical* and *group(X,Y,S)=true*, since we are interested to find rules predicting both when to split horizontally/vertically a column/section and when to group two columns/sections. No rule is generated for the case *split(X,S)=no_split* and *group(X,Y,S)=false*.

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 as well as 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. In this work the application of a first-order logic language has been explored. In this language, unary function symbols, called *attributes*, are used to describe properties of a single layout component (e.g., height and width), while binary predicate and function symbols, called *relations*, are used to express spatial relationships among layout components (e.g., part_of and on_top). An example of a training observation automatically generated by WISDOM++ follows:

```
split(c1,s)=horizontal, group(s1,s2,s)=false,
split(s1,s)=no_split, split(s2,s)=no_split ←
   step(s)=1,
   type(s1)=section, type(s2)=section, type(c1)=column,
   width(s1)=552, width(s2)=552, width(c1)=552,
   height(s1)=8, height(s2)=723, height(c1)=852,
   x_pos_centre(s1)=296, x_pos_centre(s2)=296,
   x_pos_centre(c1)=296,
```

[1] The origin of the coordinate system is at the top left-hand corner; the abscissa increases from the leftmost to the rightmost column, while the ordinate increases from the uppermost to the lowest row.

```
y_pos_centre(s1)=22, y_pos_centre(s2)=409,
y_pos_centre(c1)=426,
on_top(s1,s2)=true,
part_of(c1,s1)=true, part_of(c1,s2)=true,
no_blocks(s1)=2, no_blocks(s2)=108, no_blocks(c1)=110,
per_text(s1)=100, per_text(s2)=83, per_text(c1)=84.
```

This is a multiple-head ground clause, which has a conjunction of literals in the head. It describes the first correction applied to a page layout, where two sections and one column were originally found (Figure 1). The horizontal splitting of the column is the first correction performed by the user (Figure 2), as described by the first literal, namely *step(s)=1*. This column 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 text blocks, enclosed by the column, is 84%. It is noteworthy that the multiple-head clause above also reports that the two sections *s1* and *s2* should be neither split (literals *split(s1,s)=no_split* and *split(s2,s)=no_split*) nor grouped (literal *group(s1,s2,s)=false*) at the first correction step. Many other literals, such as *group(c1,s2,s)=false*, *group(s1,c1,s)=false*, and *group(c1,c1,s)=false*, have not been generated, since they do not represent admissible groupings according to the two constraints specified above.

Rules for the automated correction of the layout analysis can be automatically learned by means of a first-order learning system. In this work, the learning system ATRE has been used [9]. It solves the following learning problem:

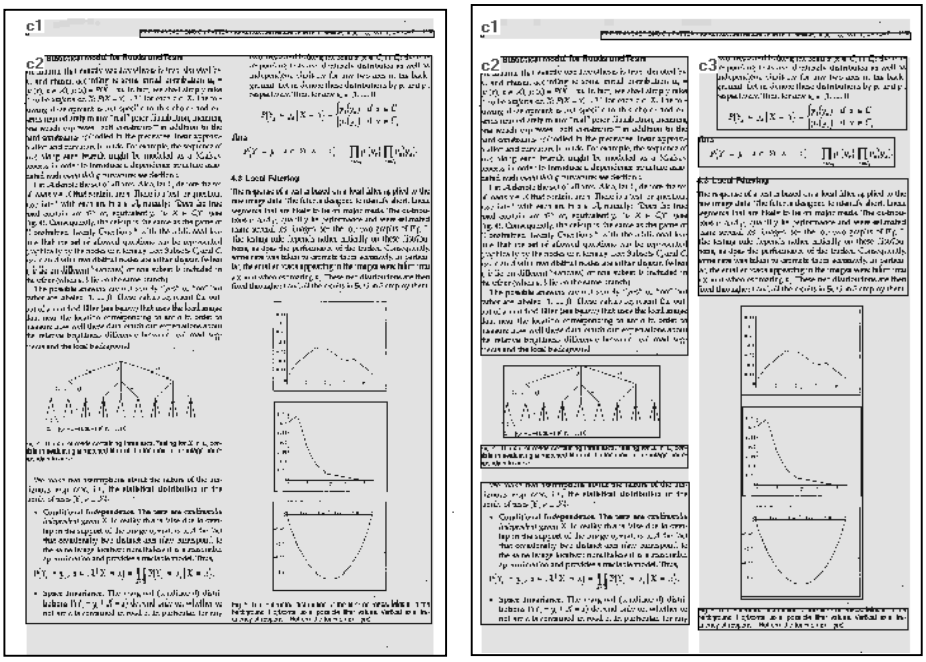

Fig. 2. Horizontal split of the column (*left*) and vertical split of column c2 (*right*). The result of the layout analysis process is in the background.

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 described 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,S)=horizontal*, *split(X,S)=vertical*, *group(X,Y,S)=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,S)=no_split* and *group(X,Y,S)=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 [7].

The preference criterion PC is a set of conditions used to discard some solutions and favor others. 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 thirteen papers published as either regular or short, in the IEEE Transactions on Pattern Analysis and Machine Intelligence, issues of January and February 1996. Each paper is a multi-page document; therefore we processed 109 document images in all, which were used for the training phase. The distribution of pages used for training purposes is reported in Table 1.

The number of training observations for ATRE corresponds to the final, corrected layout of each page (i.e., 109), plus the number of intermediate global layout structures, which are subject to corrections (i.e., 106). The total number of examples in the 215 training observations is 7786, which corresponds to the total number of literals in the multiple-head clauses. Given the set of concepts to be learned, only 106 out of 7786 examples are positive, which correspond to actual corrective actions performed by the user (vertical/horizontal splitting or grouping). The average number

of corrections performed by the user is 0.97 (i.e., 106/109) per 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.

Table 1. Training set: Distribution of pages and examples per document.

Name of the multi-page document	*No. of pages*	*No. of horizontal splits*	*No. of vertical splits*	*No. of groupings*	*Total no. of examples*
TPAMI1	14	6	5	4	1004
TPAMI2	8	4	5	0	374
TPAMI5	6	1	3	0	402
TPAMI6	2	0	0	1	83
TPAMI7	7	0	0	1	328
TPAMI8	6	2	1	2	526
TPAMI9	5	1	1	0	114
TPAMI14	10	3	4	12	1035
TPAMI15	15	9	10	0	806
TPAMI16	14	1	4	2	965
TPAMI18	10	2	8	4	1464
TPAMI22	5	2	2	0	181
TPAMI23	7	3	2	1	504
Total (training)	109	34	45	27	7786

ATRE generated a theory with 44 clauses: 19 for vertical split, 11 for horizontal split and 14 for grouping. Some clauses for the three concepts are reported below:

```
1.   split(X1,S)=horizontal ← width(X1)∈[540..567],
         height(X1)∈[848..875], step(S)∈[1..1]
2.   split(X1,S)=vertical ← width(X1)∈[536..581],
         on_top(X1,X2)=true, x_pos_centre(X1)∈[467..467],
         step(S)∈[1..1]
3.   group(X1,X2,S)=true ← width(X1)∈[408..513],
         type(X1)=column, step(S)∈[1..6], type(X2)=column
```

The interpretation of these clauses is straightforward. The first clause states that «at the first correction step, columns/areas with width between 540 and 567 pixels and height between 848 and 875 pixels should be horizontally split». The second clause states that «at the first correction step, columns/areas with a width between 536 and 581 pixels, the baricentre at point 467 on the x axis and below another column/area should be vertically split». Finally, the third clause states that «at any step between 1 and 6, two columns can be grouped if the left one[2] has a width between 408 and 513». It is noteworthy that the second clause involves the relation *on_top* and could be generated only by learning systems that operate on first-order logic descriptions, such as ATRE.

From the examples above, it is evident that some of the induced clauses (e.g., the second) are clearly specific and have been generated by the system to explain a limited number of examples (sometimes only one). Specificity of clauses is due to

[2] In this case the area is necessarily a column, since users can only group two columns or two sections.

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. However, other clauses generated by ATRE are quite general, such as the first example above.

WISDOM++ uses the induced rules to automatically correct a page layout every time a document image is processed. This operation is quick and totally transparent to the user. Data on the test set are reported in Table 2. They refer to ten additional papers published in the issues of January and February 1996 of the IEEE Transactions on Pattern Analysis and Machine Intelligence. Results of the test examples are reported in Table 3. *Omission* errors occur when correct actions on page layout are missed, while *commission* errors occur when wrong actions are "recommended" by a rule. In the case of horizontal (vertical) split, the number of possible commission errors, that is, 3189 (3200), is the sum of the number of examples of vertical (horizontal) split plus the number of no split, that is, 3153. In the case of grouping, possible commission errors equals the number of examples of *grouping(X,Y,S)=false*.

Table 2. Testing set: Distribution of pages and examples per document.

Name of the multi-page document	*No. of pages*	*No. of horizontal splits*	*No. of vertical splits*	*No. of groupings*	*Total no. of examples*
Total (testing)	109	47	36	12	7376

Table 3. Commission and omission errors performed by rules of various concepts.

Rule for	*No. omission errors*	*No. commission errors*
split(X,S)=horizontal	18/47	5/3189
split(X,S)=vertical	10/36	5/3200
grouping(X,Y,S)=true	10/12	14/4128

Unfortunately, the induced set of clauses missed most of the grouping operations, whereas it was able to correct some page layouts by performing horizontal and vertical splitting. The percentage of commission errors is very low, whereas the percentage of omission errors is quite high. This confirms our comments on the specificity of part of the learned theory, due to the reduced number of training observations with respect to the complexity of the learning task. It is also noteworthy that most of the errors occurred in few pages, where the correction process was quite complex.

5 Conclusions

This work presents a preliminary application of machine learning techniques to the problem of correcting the result of the global layout analysis process in WISDOM++. The proposed approach is alternative to inducing the complete layout structure from a set of training examples The learning problem to be solved has been introduced and the first-order logic representation of the corrections performed by the user has been illustrated. Experimental results on a set of multi-page documents showed that the proposed approach is able to capture relatively simple layout corrections. Inaccuracy for complex processes can be mainly attributed to the limited size of training

documents. A more extensive experimentation is planned to confirm these initial conclusions. A further research issue to be investigated concerns the application of a learning system like ATRE, devised to solve classification problems, to a typical planning task. Finally, we intend to investigate the problem of incrementally refining the set of rules generated by ATRE, when new training observations are made available.

Acknowledgments

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)

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), pp. 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, pp. 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, pp. 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, pp. 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, pp. 466- 471, 1995.
6. Esposito F., Malerba D., & Lisi F.A.: Machine learning for intelligent processing of printed documents, *Journal of Intelligent Information Systems*, 14(2/3), pp. 175-198, 2000.
7. 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, pp. 93-111, Springer: Berlin, 2000.
8. 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.
9. Malerba D., Esposito F., & Lisi F.A.: Learning recursive theories with ATRE, *Proc. of the 13th European Conf. on Artificial Intelligence*, John Wiley & Sons, pp. 435-439, 1998.
10. Srihari S.N., & Zack G.W.: Document Image Analysis. *Proc. of the 8th Int. Conf. on Pattern Recognition*, pp. 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, pp. 243-247, 1997.