Semantic-Based Access to Digital Document Databases

F. Esposito, S. Ferilli, T.M.A. Basile, and N. Di Mauro

Dipartimento di Informatica, University of Bari, Italy
{esposito, ferilli, basile, nicodimauro}@di.uniba.it

Abstract. Discovering significant meta-information from document collections is a critical factor for knowledge distribution and preservation. This paper presents a system that implements intelligent document processing techniques, by combining strategies for the layout analysis of electronic documents with incremental first-order learning in order to automatically classify the documents and their layout components according to their semantics. Indeed, an in-deep analysis of specific layout components can allow the extraction of useful information to improve the semantic-based document storage and retrieval tasks. The viability of the proposed approach is confirmed by experiments run in the real-world application domain of scientific papers.

1 Introduction

Since having documents in electronic form makes their management significantly easier, much research in the last years looked for approaches to handle the huge amount of legacy documents in paper format according to the semantics of their components [8]. Conversely, almost all documents nowadays are generated directly in digital format, and stored in distributed repositories whose main concerns and problems consist in the acquisition and organization of the information contained therein. Manually creating and maintaining an updated index is clearly infeasible, due to the potentially huge amount of data to be handled, tagged and indexed. Hence a strong motivation for the research concerned with methods that can provide solutions for automatically acquiring new knowledge.

This paper deals with the application of intelligent techniques to the management of a collection of scientific papers on the Internet, aimed at automatically extracting from the documents significant information, useful to properly store and retrieve them. In this application domain, to identify the subject and context of a paper, an important role is played by components such as title, authors, abstract and bibliographic references. Three processing stages are typically needed to identify a document significant components: Layout Analysis, Document Classification and Document Understanding. We propose to exploit Machine Learning techniques to carry out the last two steps. In particular, the need for expressing relations among layout components requires the use of symbolic first-order techniques, while the continuous flow of new document calls for *incremental* abilities that can revise a faulty knowledge previously acquired.

M.-S. Hacid et al. (Eds.): ISMIS 2005, LNAI 3488, pp. 373–381, 2005.

In the following we present DOMINUS, a system that can extract the layout structure from electronic documents and applies incremental symbolic learning techniques to tag them.

2 DOMINUS: Document Analysis

The availability of large, heterogeneous repositories of electronic documents is increasing rapidly, and the need for flexible, sophisticated document manipulation tools is growing correspondingly. These tools can benefit by exploiting the *logical structure*, a hierarchy of visually observable organizational components of a document, such as paragraphs, lists, sections, etc. While Document analysis is traditionally concerned with scanned images, in this paper we take into account electronic documents, whose advantages over paper ones include compact and lossless storage, easy maintenance, efficient retrieval and fast transmission.

Some of the major advantages of digital documents are their explicit structure, possibly described by a hierarchy of physical (columns, paragraphs, textlines, words, tables, figures, etc.) or logical (titles, authors, affiliations, abstracts, etc.) components, or both. This structural information can be very useful in indexing and retrieving the information contained in the document. Physical layout and logical structure analysis of document images is a crucial stage in a document processing system. In particular, we focus on documents in PostScript (PS) or Portable Document Format (PDF), and propose a new approach, based primarily on layout information, for discovering a full logical hierarchy.

2.1 Basic PostScript Analysis

PostScript [1] is a simple interpretative programming language with powerful graphical capabilities. Its primary application is to describe the appearance of text, graphical shapes, and sampled images on printed or displayed pages. PDF is an evolution of PostScript rapidly gaining acceptance as a promising file format for electronic documents. Like PostScript, it is an open standard, enabling integrated solutions from a broad range of vendors.

Our layout analysis process uses a preprocessing algorithm, named pstopl, that takes as input a PDF or PS document (such as the PDF version of this paper) and transforms it into its corresponding *XML basic representation*, that describes the initial digital document as a set of pages, each of which is composed of *basic blocks*. It uses Ghostscript, a software package that provides a PostScript and PDF interpreter, able to convert PostScript files to many raster formats, display them on screen, and print them. Specifically, pstopl performs a syntactic transformation from PostScript to XML. A similar algorithm is pstoedit [6], that translates PostScript and PDF graphics into other vector formats. More technically, pstopl rewrites basic PostScript operators by turning each instruction into an object `id`, described by means of the following predicates:

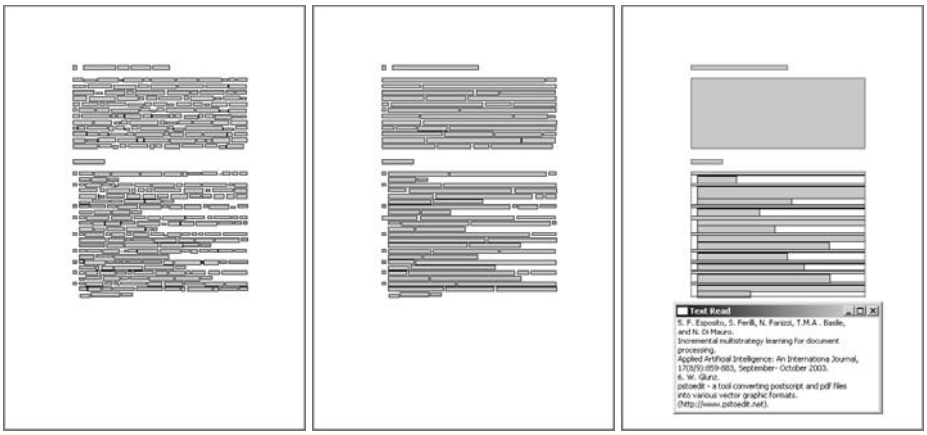

Fig. 1. Word, line and final layout analysis representations automatically extracted from the last page of *this paper*, plus an excerpt from the references

`box(id, XY0, XY1, font, dimension, RGB, row, text)` for the text;
`stroke(id, XY0, XY1, RGB, thickness)` for the lines;
`fill(id, XY0, XY1, RGB)` for the Monochromat Areas;
`image(id, XY0, XY1)` for the Raster Image;
`page(n, w, h)` for the Pages.

where `XY0` and `XY1` are the top-left and bottom-right coordinates of the bounding box respectively; `font` and `dimension` are the font-type and size of the text; `RGB` is the color of the bounding box content in #rrggbb format; `row` is the number of the row in the document where the bounding box is placed; `text` is the text contained; `thickness` is referred to the lines; `n`, `w` and `h` are, respectively, the number, width and height of the page. For example, the PostScript instruction to display a text becomes an object describing a text with attributes for the geometry (location on the page) and appearance (font, colour, etc.) as reported in the following excerpt of a `pstopl` output.

```
box(1,108.3,753.2,176.5,738.3,Times-Bold,28,#000000,738,"Notions").
stroke(14, 63.7178, 189.721, 230.279, 189.721, #010101, 6).
fill(1173, 305.124, 172.98, 550.201, 172.184, #000000).
image(13, 92.76, 280.561, 204.84, 195.841).
page(1, 595, 841).
```

2.2 Geometric Layout Analysis

Objects in the layout of a document are spatially organized in *frames*, defined as collections of objects completely surrounded by white space. There is no exact correspondence between the layout notion of a frame and a logical notion such as a *paragraph*: two columns on a page correspond to two separate frames, while a paragraph might begin in one column and continue into the next one.

DOMINUS, after transforming the original document into its basic XML representation, performs the layout analysis of the document by applying an efficient and easy-to-implement algorithm named doc, a variant of that reported in [3] for addressing the geometric layout analysis problem. doc analyzes the whitespace and background structure of documents in terms of rectangular covers. The output of doc is the *XML layout structure* (see Figure 1) of the original document, obtained through a process that is not a merely syntactic transformation from PS/PDF to XML. As reported in [3], given a collection of rectangles C in the plane, all contained within some given bounding rectangle r_b, the *maximal white rectangle problem* is to find a rectangle $\overline{r} \subset r_b$ that maximizes a function $Q(\cdot)$ among all the possible rectangles $r \subseteq r_b$, where r overlaps none of the rectangles in C, and Q satisfies the condition $r \subseteq r' \Rightarrow Q(r) \leq Q(r')$ for any two rectangles r and r' (in our case, Q is chosen as the area of the rectangle).

After the background of the document has been identified using doc, it is possible to compute its complement, thus obtaining the desired output consisting of two levels of description. The former refers to single blocks filled with the same kind of content, the latter consists in rectangular frames each of which may be made up of many blocks of the former type. Thus, the overall description of the document includes both kinds of objects, plus information on which frames include which blocks and on the actual spatial relations between frames and between blocks in the same frame (e.g., above, touches, etc.). This allows to maintain both levels of abstraction independently.

The following improvements of the basic algorithm reported in [3] were introduced in doc as reported in Algorithm 2.2:

Lines: line segments in a document are considered first, and divide the document layout into two parts.

Reducing Basic Blocks: to improve the efficiency of doc, two preliminary phases are applied to the XML basic representation. The basic blocks identified by pstopl often correspond to fragments of words, so that a first aggregation based on their overlapping or adjacency is needed in order to obtain blocks surrounding whole words. A further aggregation could be performed, based on proximity, to have blocks that group words in lines (see Figure 1). This yields the *XML line-level description* of the document, that contains fewer blocks than the XML basic description and represents the actual input to the document layout analysis algorithm based on whitespace and background structure analysis.

Rejecting Small White Rectangles: doc uses a threshold, α, on the dimensions and/or area of the rectangles to be found in order to avoid the extraction of non-significant backgrounds such as inter-word or inter-line spaces.

Stop Criterion: since doc finds iteratively the maximal white rectangles, a stop criterion was needed. It was empirically defined as the moment in which the area of the new white rectangle represents a percentage of the total white area in the document (computed by subtracting the sum of all the areas of the basic blocks from the whole area of the document) less than a given threshold, β.

In the resulting XML description of the original document obtained by DOMINUS, each logical component has been identified and described (by means of its geometric attributes and relationships with respect to other components) and can be tagged according to the sematic role it plays into the document (e.g., title, abstract, author, etc.).

Algorithm 1 Pseudo-code of the optimization of the Breuel Algorithm

function findLayout(M: bound (page's margin); O: obstacles; α, β: float) : WhiteRectangles: Set of Rectangles

```
group_blocks_in_words(O,O'); /* Identify blocks surrounding whole words */
group_words_in_lines(O',O"); /* Identify blocks grouping words in lines */
enqueue(C,(M,O")); /* C is a priority queue based on obstacles' area */
WhiteArea := area(M) - area(O"); percIncr := 1;
while not isempty(C) and (percIncr > β) do
  (B,Obstacles) = dequeue(C);
  percIncr := area(b) / WhiteArea;
  if (obstacles == ∅ and area(B) > α) then
    WhiteRectangles := WhiteRectangles ∪ {B};
    filter(C,B); /* Split each element of C that overlaps B by using B as a pivot */
  else
    p = pickpivot(obstacles); /* The lines are a preferential pivot */
    b0 = (p.x1,b.y0,b.x1,b.y1), b1 = (b.x0,b.y0,p.x0,b.y1)
    b2 = (b.x0,p.y1,b.x1,b.y1), b3 = (b.x0,b.y0,b.x1,p.y0)
    for i := 0 .. 3 do
      if ( area(bi) > 0 and bi is not included in any bound of C) then
        sub_obstacles = {u ∈ obstacles | u overlaps bi}
        enqueue(C,(bi,sub_obstacles))
Return WhiteRectangles
```

3 DOMINUS: Document Classification/Understanding

The need of automatically labelling the huge amount of documents in a Digital Library environment, along which their significant components, in order to help the automatic organization of the document collection for a more efficient storage and retrieval process, suggested the use of a concept learning system to infer rules for such task. One dimension along with concept learning algorithms can be characterized is whether they operate in an incremental or non-incremental way. The former revise the current concept definition in response to each newly observed training instance, if necessary. The latter infer concepts requiring the whole set of the training instances to be available at the beginning of the learning process. The appropriateness of these two approaches has to be evaluated according to the learning task at hand: in the case of evolutionary environments in which the instances/information come sequentially, as in digital documents collections, an incremental algorithm is preferable since revising an existing hypothesis is more efficient and less expensive than generating a hypothesis from scratch each time a new instance is observed. Furthermore, the purpose of discovering significant

knowledge to be used as meta-information for the content-based retrieval and management of document collections cannot leave aside the complexity of the domain, due to the great number of interesting layout components, and their relations, to be discovered in the documents. This suggests the exploitation of symbolic first-order logic as a powerful representation language to handle such a situation. Based on these considerations, we decided to adopt INTHELEX [5] as a system to learn rules for the automatic identification of logical components.

INTHELEX is an incremental Inductive Logic Programming [7] system able to induce conceptual descriptions from positive and negative examples. Specifically, the hypotheses are represented as sets of first-order constant-free clauses whose body describes the definition of the concept in the head. The system incorporates two refinement operators to restore the correctness of the theory, one for generalizing hypotheses that reject positive examples, and the other for specializing hypotheses that explain negative examples. Furthermore, whenever a new example is taken into account the system checks if any sub-concept can be recognized by deduction in its description according to the definitions learned thus far and the background knowledge, in which case the information concerning those concepts is added (properly instantiated) to the example description. Another peculiarity of the system is that it is able to exploit multistrategy operators for shifting the representation language (abstraction) and/or to hypothesize unseen information (abduction).

In order to exploit INTHELEX, a suitable first-order logic representation of the document must be provided. Thus, once the layout components of a document are automatically discovered as explained in the previous section, the next step concerns the description of the pages, blocks and frames according to their size, spatial [9] and membership relations. Dealing with multi-page documents, the document description must be enriched with *page information* such as: the page number; whether the page is at the beginning, in the middle or at the end of the document; whether the page is the last one; the total number of pages in a document. As pointed out, the layout analysis process ends with a set of rectangles in which each single page is divided. Such rectangles are described by means of their type (text, graphic, line) and their horizontal and vertical position in the document. Furthermore, the algebraic relations $\subset$ and $\supset$ are exploited to express the inclusion between pages and frames and between blocks and frames. The other relation that can be described between rectangles is the spatial one. It is applied to all the blocks belonging to the same frame and to all the adjacent frames (given a rectangle r, the set of rectangles A that are adjacent to r is made up of all the $r_i \in A$ such that r_i is the nearest rectangle of r in the same plane). Fixed a rectangle r, one can ideally divide the plane containing it in 25 parts, as reported in Figure 2. Then, the relation between r and the other rectangles is described in terms of the occupied planes with respect to r. Moreover, such kind of representation of the planes allows the introduction in the example description of the topological relations [4, 9], including closeness, intersection and overlapping between rectangles. However, the topological information can be deduced

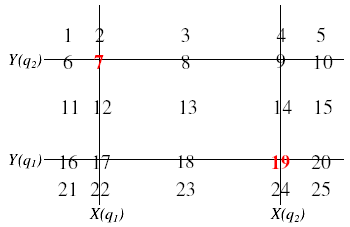

Fig. 2. Representation Planes for a rectangle according to [9]

by the spatial relationships, and thus it can be included by the learning system during the learning process exploiting deduction and abstraction.

In order to evaluate the proposed document processing technique, some experiments were designed based on a dataset made up of 108 documents (scientific papers) coming from online repositories. Specifically, 31 papers were formatted according to the Springer-Verlag `LNCS` style, 32 according to the `Elsevier` journals one, 20 according to the `ICML` proceedings style and other 25 according to the `ECAI` proceedings style. Each document was pre-processed and automatically described in a first-order logic language, along the features above mentioned. Each document instance obtained in such a way was considered a positive example for the class it belongs to, and as a negative example for all the other classes to be learned. The system performance was evaluated according to a 10-fold cross-validation methodology, ensuring that all the learning and test sets contained the same proportion of positive and negative examples. Furthermore, the system was provided with background knowledge expressing topological relations, and abstraction was exploited to discretize numeric values of size and position. In the following, some examples of both are given.

Fragment of background knowledge for topological relations:

```
over_alignment(B1,B2):-
   occupy_plane_9(B1,B2),not(occupy_plane_4(B1,B2)).
over_alignment(B1,B2) :-
   occupy_plane_10(B1,B2), not(occupy_plane_5(B1,B2)).
left_alignment(B1,B2) :-
   occupy_plane_17(B1,B2), not(occupy_plane_16(B1,B2)).
left_alignment(B1,B2) :-
   occupy_plane_22(B1,B2), not(occupy_plane_21(B1,B2)).
touch(B1,B2) :-
   occupy_plane_17(B1,B2),not(occupy_plane_13(B1,B2)).
touch(B1, B2) :-
   occupy_plane_18(B1,B2),not(occupy_plane_13(B1,B2)).
```

Extract of abstraction theory for rectangles width discretization:

```
width_very_small(X):-
    rectangle_width(X,Y),
    Y >= 0,Y =< 0.023.

width_small(X):-
    rectangle_width(X,Y),
    Y > 0.023,Y =< 0.047.
```

Table 1. System Performance on the Classification Phase

Class	Revisions	RunTime (sec.)	Accuracy %
LNCS	15	165.70	93.1
ICML	8.5	51.86	98
Elsevier	9.8	118.34	98
ECAI	11	3108.98	97

Table 2. System Performance on the Understanding Phase

LNCS	Revisions	RunTime (sec.)	Accuracy %
title	11.29	33.467	95.93
author	12.27	47.88	95.39
abstract	16.48	133.706	93.06
references	13.29	50.94	95.24
ICML	Revisions	RunTime (sec.)	Accuracy %
title	12.13	51.668	97.87
author	11.7	29.07	97.12
abstract	10.73	111.130	97.51
references	11.17	76.224	97.54

```
width_medium_small(X):-
    rectangle_width(X,Y),
    Y > 0.047,Y =< 0.125.
```

```
width_medium(X):-
    rectangle_width(X,Y),
    Y > 0.125,Y =< 0.203.
```

In general, due to the different layout components that could be interesting for a specific class of documents but not for others, the image understanding phase must be preceded by a classification process in order to recognize the correct class the document belongs to. Hence, a first experiment on the inference of rules for the classification step was run, showing good system performance in terms of execution time, predictive accuracy and number of theory revisions, as reported in Table 1. The lowest accuracy refers to LNCS, which could be expected due to the less strict check for conformance to the layout standard by the editor. The worst runtime for the ECAI class can be explained by the fact that this is actually a generalization of ICML (from which it differs because of the absence of two horizontal lines above and below the title), which makes hard the revision task for the learning system, as confirmed by the greater number of revisions needed to learn the former with respect to the latter. Anyway, the high predictive accuracy for this class should ensure that few revisions will be needed. Moreover, taking into account that each learning set was made up of 97 examples, an average of 11 revisions can be in any case considered a positive results, since only one revision every 6.5 examples is needed, on average, in the worst case (LNCS). A second experiment concerning the image understanding phase, aimed at learning rules able to identify the layout components, was performed on the *title*, *authors*, *abstract* and *references* layout components of the documents belonging to the LNCS and ICML documents. These two classes

were chosen since they represent two distinct kinds of layout (a single-column the former, a double-column the latter). In Table 2 the averaged results of the 10 folds along with the execution time, predictive accuracy and number of theory revisions are reported. As it is possible to note, also in this experiment the system showed good performance, specifically as regards the references on which we intend to base additional processing steps.

4 Conclusion and Future Work

This paper presented a proposal for embedding intelligent techniques in the electronic documents processing task. Specifically, incremental first-order learning strategies are exploited for automatically classifying the documents and their layout components according to their semantics. In particular, we focused on the domain of scientific papers, for which we believe that an in-deep analysis of specific layout components, such as the bibliographic references, can improve the identification of the subject and context of the paper. In this respect, we are currently working on the improvement of some initial research already present in the literature [2] and on the development of new, knowledge-based ones. More future work will concern the improvement of layout analysis algorithm, by developing more effective methods for basic-block aggregation and by defining flexible parameters based on the structure of the specific document under processing.

References

1. Adobe Systems Incorporated. *PostScript language reference manual – 2nd ed.* Addison Wesley, 1990.
2. D. Besagni and A. Belaid. Citation recognition for scientific publications in digital libraries. In *Proceedings of the 1st International Workshop on Document Image Analysis for Libraries (DIAL 2004), 23-24 January 2004, Palo Alto, CA, USA*, pages 244–252. IEEE Computer Society, 2004.
3. T.M. Breuel. Two geometric algorithms for layout analysis. In *Workshop on Document Analysis Systems*, 2002.
4. M. Egenhofer. Reasoning about binary topological relations. In Gunther O. and Schek H.-J., editors, *Second Symposium on Large Spatial Databases*, volume 525 of *LNCS*, pages 143–160. Springer, 1991.
5. F. Esposito, S. Ferilli, N. Fanizzi, T.M.A. Basile, and N. Di Mauro. Incremental multistrategy learning for document processing. *Applied Artificial Intelligence: An Internationa Journal*, 17(8/9):859–883, September-October 2003.
6. W. Glunz. pstoedit - a tool converting postscript and pdf files into various vector graphic formats. (http://www.pstoedit.net).
7. S. Muggleton and L. De Raedt. Inductive logic programming: Theory and methods. *Journal of Logic Programming*, 19/20:629–679, 1994.
8. G. Nagy. Twenty years of document image analysis in PAMI. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(1):38–62, 2000.
9. D. Papadias and Y. Theodoridis. Spatial relations, minimum bounding rectangles, and spatial data structures. *International Journal of Geographical Information Science*, 11(2):111–138, 1997.