Machine Learning for Digital Document Processing: From Layout Analysis To Metadata Extraction

Floriana Esposito[1], Stefano Ferilli[1], Teresa M.A. Basile[1], and Nicola Di Mauro[1]

Università degli Studi di Bari
Dipartimento di Informatica
Via Orabona, 4
70126 Bari - Italy
{esposito,ferilli,basile,ndm}@di.uniba.it

Summary. In the last years, the spread of computers and the Internet caused a significant amount of documents to be available in digital format. Collecting them in digital repositories raised problems that go beyond simple acquisition issues, and cause the need to organize and classify them in order to improve the effectiveness and efficiency of the retrieval procedure. The success of such a process is tightly related to the ability of understanding the semantics of the document components and content. Since the obvious solution of manually creating and maintaining an updated index is clearly infeasible, due to the huge amount of data under consideration, there is a strong interest in methods that can provide solutions for automatically acquiring such a knowledge. This work presents a framework that intensively exploits intelligent techniques to support different tasks of automatic document processing from acquisition to indexing, from categorization to storing and retrieval.

The prototypical version of the system DOMINUS is presented, whose main characteristic is the use of a Machine Learning Server, a suite of different inductive learning methods and systems, among which the more suitable for each specific document processing phase is chosen and applied. The core system is the incremental first-order logic learner INTHELEX. Thanks to incrementality, it can continuously update and refine the learned theories, dynamically extending its knowledge to handle even completely new classes of documents.

Since DOMINUS is general and flexible, it can be embedded as a document management engine into many different Digital Library systems. Experiments in a real-world domain scenario, scientific conference management, confirmed the good performance of the proposed prototype.

1 Introduction & Motivations

In the World Wide Web era, a huge amount of documents in digital format are spread throughout the most diverse Web sites, and a specific research area, focused on principles and techniques for setting up and managing document collections in a digital form, quickly expanded. Usually, this large repositories of digital documents are defined as *Digital Libraries*, intended as distributed collections of textual and/or multimedia documents, whose main goal is the acquisition and the organization of the information contained therein.

During the past years a considerable effort was spent in the development of intelligent techniques in order to automatically transform paper documents into digital format, saving the original layout with the aim of reconstruction. Machine Learning techniques have been applied to attain this goal, and a successful applications in preserving cultural heritage material is reported in [1].

Today, most documents are generated, stored and exchanged in a digital format, although it is still necessary to maintain the typing convention of classical *paper* documents. The specific problem we will deal with consists in the application of intelligent techniques to a system for managing a collection of digital documents on the Internet; such a system, aimed at automatically extracting significant information from the documents, is useful to properly store, retrieve and manage them in a *Semantic Web* perspective [2]. Indeed, organizing the documents on the grounds of the knowledge they contain is fundamental for being able to correctly access them according to the user's particular needs. For instance, in the scientific papers domain, in order to identify the subject of a paper and its scientific context, an important role is played by the information available in components such as Title, Authors, Abstract and Bibliographic references. This last component in particular, with respect to others, is a source of problems both because it is placed at the end of the paper, and because it is, in turn, made up of different sub-components containing various kinds of information, to be handled and exploited in different ways.

At the moment we are not aware of techniques able to automatically annotate the layout components of digital documents, without reference to a specific template. We argue that a process is still necessary to identify the significant components of a digital document through three typical phases: Layout Analysis, Document Image Classification and Document Image Understanding. As widely known, *Layout Analysis* consists in the perceptual organization process that aims at identifying the single blocks of a document and at detecting relations among them (*Layout Structure*); then, associating the proper logical role to each component yields the *Document Logical Structure*. Since the logical structure is different according to the kind of document, two steps are in charge of identifying such a structure: *Document Image Classification*, aiming at the categorization of the document (e.g., newspaper, scientific paper, email, technical report, call for papers) and *Document Image Understanding*, aiming at the identification of the significant layout components for that class. Once the class as been defined it is possible to associate to each component a tag that expresses its role (e.g., signature, object, title, author, abstract, footnote, etc.). We propose to apply multistrategy Machine Learning techniques along these phases of document processing where the classical statistical and numerical approaches to classification and learning may fail, being not able to deal with the lack of a strict layout regularity in the variety of documents available online.

The problem of Image Document Processing requires a first-order language representation for two reasons. First, classical attribute-value languages describe a document by means of a fixed set of features, each of which takes a value from a corresponding pre-specified value set; the exploitation of this language in this domain represents a limitation since one cannot know *a priori* how many components make up a generic document. Second, in attribute-value formalism it is not possible to represent and efficiently handle the relationships among components; the information coming from the topological structure of all components in a document turns out to be very useful in document understanding. For instance, in a scientific paper, it is useful to know that the acknowledgments usually appear above the references section and in the end of the document, or that the affiliation of the authors is reported generally at the beginning of the document, below or on the right of their names.

The continuous flow of new and different documents in a Web repository or in Digital Libraries calls for *incremental* abilities of the system, that must be able to update or revise a faulty knowledge previously acquired for identifying the logical structure of a document. Traditionally, Machine Learning methods that automatically acquire knowledge in developing intelligent systems, require to be provided with a set of training examples, belonging to a defined number of classes, and exploit them altogether in a batch way.

Although sometimes the term *incremental* is used to define some learning method [3, 4, 5, 6], incrementality generally refers to the possibility of adjusting some parameters in the model on the grounds of new observations that become available when the system is already operational. Thus, classical approaches require that the number of classes is defined and fixed since the beginning of the induction step: this prevents the opportunity of dealing with totally new instances, belonging to new classes, that require the ability to incrementally revise a domain theory as soon as new data are encountered. Indeed, Digital Libraries require autonomous or semi-autonomous operation and adaptation to changes in the domain, the context, or the user needs. If any of these changes happens, the classical approach requires that the entire learning process is restarted to produce a model capable of coping with the new scenario. Such requirements suggest that incremental learning, as opposed to classical batch one, is needed whenever either incomplete information is available at the time of initial theory generation, or the nature (and the kinds) of the concepts evolves dynamically. E.g., this is the case of modifications in time of typing style of documents that nevertheless belong to the same class or of the introduction of a completely new class. Incremental processing allows for continuous responsiveness to the changes in the context, can potentially improve efficiency and deals with concept evolution. The incremental setting implicitly assumes that the information (observations) gained at any given moment is incomplete, and thus that any learned theory could be susceptible of changes.

This chapter presents the prototypical version of DOMINUS (DOcument Management INtelligent Universal System): such a system is characterized by the intensive exploitation of intelligent techniques in each step of document processing from acquisition to indexing, from categorization to storing and retrieval. Since it is general and flexible, it can be embedded as a document management engine into many different Digital Library systems. In the following, after a brief description of the architecture of DOMINUS, the results of the layout analysis on digital documents are discussed, with the interesting improvements achieved by using kernel-based approaches and incremental first-order learning techniques: the satisfying results in document layout correction, classification and understanding allow to start an effective structural metadata extraction. Then, the categorization, filing and indexing tasks are described with the results obtained in the effective retrieval of scientific documents. Finally, the application of the system in a real-world domain scenario, scientific conference management, is reported and discussed.

2 The Document Management System Architecture

This section briefly presents the overall architecture of DOMINUS, reported in Figure 1. A central role is played by the Learning Server, which intervenes during different processing steps in order to continuously adapt the knowledge taking into consideration new experimental evidence and changes in the context. The corresponding process flow performed by the system from the original digital documents acquisition to text extraction and indexing is reported in Figure 2.

The layout analysis process on documents in digital format starts with the application of a pre-processing module, called WINE (Wrapper for the Interpretation of Non-uniform Electronic document formats), that rewrites basic PostScript operators to turn their drawing instructions into objects (see Section 3). It takes as input a digital document and produces (by an intermediate vector format) the initial document's XML basic representation, that describes it as a set of pages made up of basic blocks. Due to the large number of basic blocks discovered by WINE, that often correspond to fragments of words, it is necessary a first aggregation based on blocks overlapping or adjacency, yielding composite blocks corresponding to whole words. The number of blocks after this step is still large, thus a further aggregation (e.g., of words into lines) is needed. Since grouping techniques based on the mean distance between blocks proved unable to correctly handle the case of multi-column documents, such a task was cast to a multiple instance problem (see Section 3.1) and solved exploiting the kernel-based method proposed in [7], implemented in a Learning Server module that is able to generate rewriting

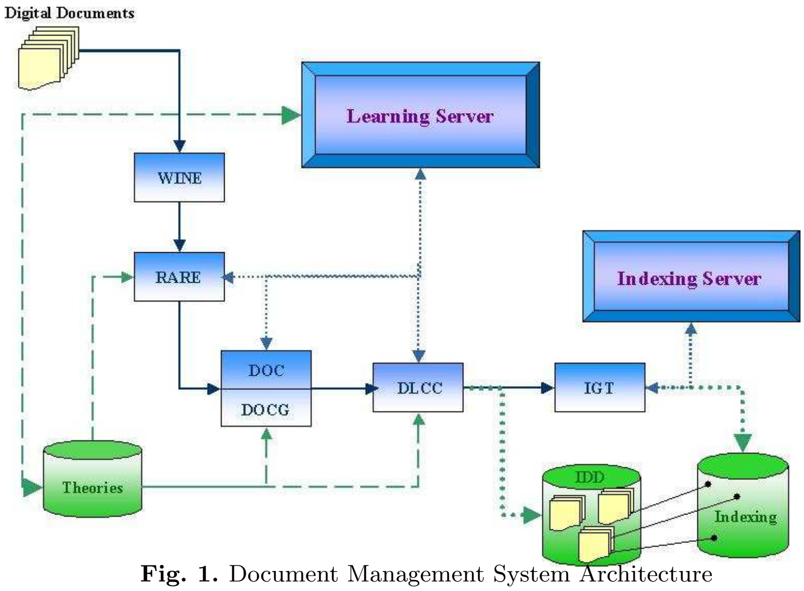


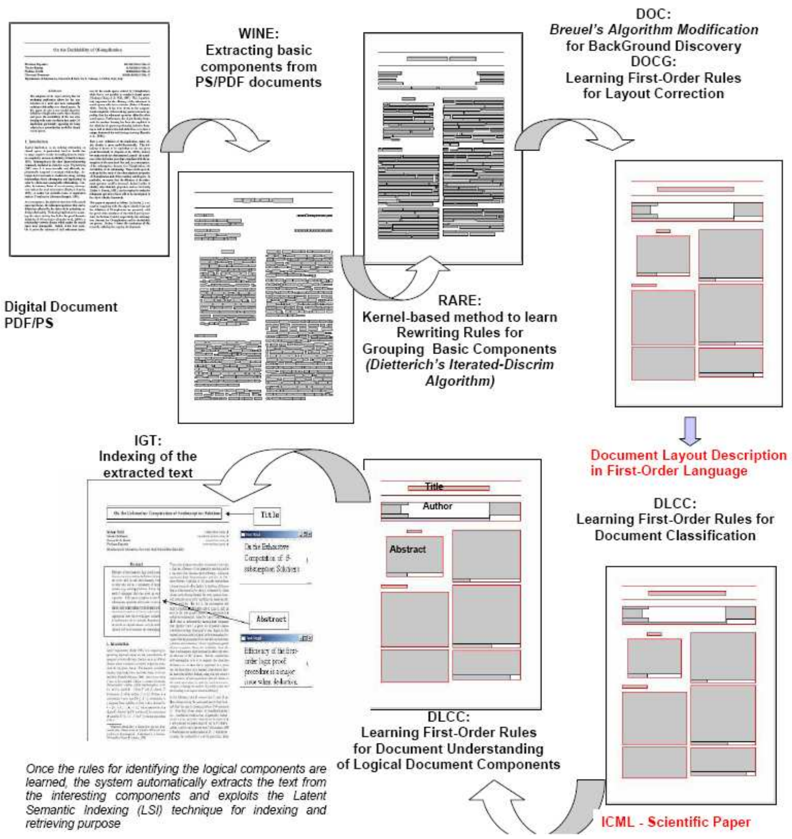
Fig. 1. Document Management System Architecture

rules that suggest how to set some parameters in order to group together word blocks to obtain lines. The inferred rules will be stored in the Theories knowledge base for future exploitation by RARE (Rule Aggregation REwriter) and modification.

Once such a line-block representation is generated, DOC (Document Organization Composer) collects the semantically related blocks into groups by identifying the surrounding frames based on white spaces and the results of the background structure analysis. This is an improvement of the original Breuel's algorithm [8], that finds iteratively the maximal white rectangles in a page: here the process is forced to stop before finding insignificant white spaces such as inter-word or inter-line ones (see Section 3.2).

At the end of this step, some blocks might not be correctly recognized. In such a case a phase of layout correction is needed, that is automatically performed in DOCG (Document Organization Correction Generator) by exploiting embedded rules stored in the Theories knowledge base. Such rules were automatically learned from previous manual corrections collected on some document during the first trials and using the Learning Server.

Once the layout structure has been correctly and definitely identified, a semantic role must be associated to each significant components in order to perform the automatic extraction of the interesting text with the aim of improving document indexing. This step is performed by DLCC (Document and Layout Components Classifier) by firstly associating the document to a class that expresses its type and then associating to every significant layout component a tag expressing its role. Both these steps are performed thanks to



Once the rules for identifying the logical components are learned, the system automatically extracts the text from the interesting components and exploits the Latent Semantic Indexing (LSI) technique for indexing and retrieving purpose

Fig. 2. Document Management System Process Flow

theories previously learned and stored in the Theories knowledge base. In case of failure these theories can be properly updated. The theory revision step is performed by a first-order incremental learning system that runs on the new observations and tries to modify the old theories in the knowledge base. At the end of this step both the original document and its XML representation, enriched with class information and components annotation, is stored in the Internal Document Database, IDD.

Finally, the text is extracted from the significant components and the Indexing Server is called by the IGT (IndexinG of Text) module to manage such information, useful for an effective content-based document retrieval.

3 Layout Structure Recognition

Based on the ODA/ODIF standard, any document can be progressively partitioned into a hierarchy of abstract representations, called its *layout structure*.

Here we describe an approach implemented for discovering a full layout hierarchy in digital documents based primarily on layout information.

The layout analysis process starts with a preliminary preprocessing step performed by a module that takes as input a generic digital document and produces a corresponding vectorial description. An algorithm for performing this task is PSTOEDIT [9], but it was discarded because it only applies to PostScript (PS) and Portable Document Format (PDF) documents and returns a description without sufficient details for our purposes.

Thus, a module named WINE (Wrapper for the Interpretation of Non-uniform Electronic document formats), has been purposely developed. At the moment, it deals with digital documents in PS or PDF formats, that represent the current *de facto* standard for document interchange. The PostScript [10] language is a simple interpretative programming language with powerful graphical capabilities that allow to precisely describe any page. The PDF [11] language is an evolution of PostScript that rapidly gained acceptance as a file format for digital documents. Like PostScript, it is an open standard, enabling integrated solutions from a broad range of vendors. In particular, WINE consists of a rewriting of basic PostScript operators that turns the instructions into objects. For example, the PS instruction to display a text becomes an object describing a text with attributes for the geometry (location on the page) and appearance (font, color, etc.). The output of WINE is a vector format describing the initial digital document as a set of pages, each of which is composed of *basic blocks*. The descriptors used by WINE for representing a document are the following:

box(id,x0,y0,x1,y1,font,size,RGB,row,string): a piece of text in the document, represented by its bounding box;

stroke(id,x0,y0,x1,y1,RGB,thickness): a graphical (horizontal/vertical) line of the document;

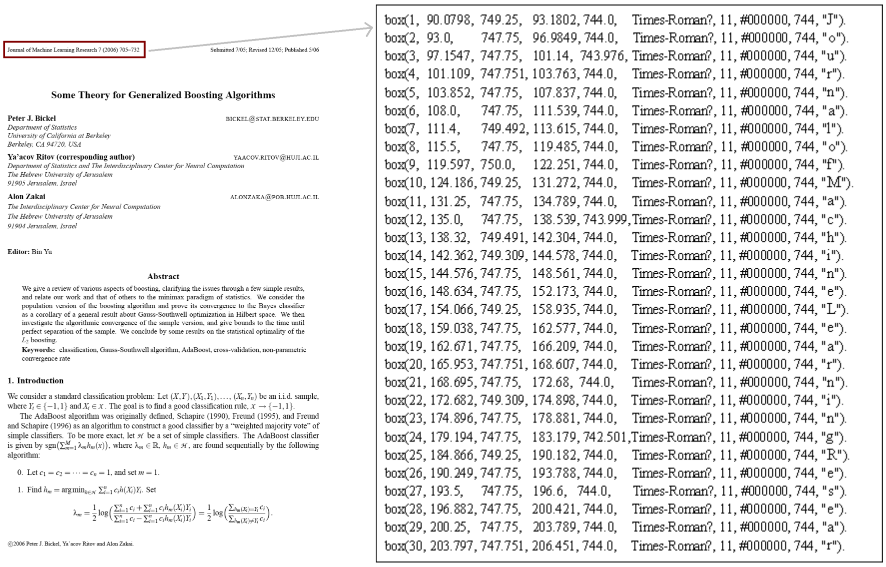
fill(id,x0,y0,x1,y1,RGB): a closed area filled with one color;

image(id,x0,y0,x1,y1): a raster image;

page(n,w,h): page information;

where: id is the block identifier; $(x0, y0)$ and $(x1, y1)$ are respectively the upper-left/lower-right coordinates of the block (note that $x0 = x1$ for vertical lines and $y0 = y1$ for horizontal lines); font is the the type font; size represents the text size; RGB is the color of the text, line or area in #rrggbb format; row is the index of the row in which the text appears; string is the text of the document contained in the block; thickness is the thickness of the line; n represents the page number; w and h are the page width and height, respectively. Figure 3 reports an extract of the vectorial transformation of the document.

Such a vectorial representation is translated into an *XML basic representation*, that will be modified as long as the layout analysis process proceeds, in order to represent the document by means of increasingly complex aggregations of basic components progressively discovered by the various layout analysis phases.



Journal of Machine Learning Research 7 (2006) 705–732 Submitted 7/05; Revised 12/05; Published 5/06

Some Theory for Generalized Boosting Algorithms

Peter J. Bickel BICKEL@STAT.BERKELEY.EDU
Department of Statistics
University of California at Berkeley
Berkeley, CA 94720, USA

Ya'acov Ritov (corresponding author) YAACOV.RITOV@HUJI.AC.IL
Department of Statistics and The Interdisciplinary Center for Neural Computation
The Hebrew University of Jerusalem
91905 Jerusalem, Israel

Alon Zakai ALONZAKA@POB.HUJI.AC.IL
The Interdisciplinary Center for Neural Computation
The Hebrew University of Jerusalem
91904 Jerusalem, Israel

Editor: Bin Yu

Abstract

We give a review of various aspects of boosting, clarifying the issues through a few simple results, and relate our work and that of others to the minimax paradigm of statistics. We consider the population version of the boosting algorithm and prove its convergence to the Bayes classifier as a corollary of a general result about Gauss-Southwell optimization in Hilbert space. We then investigate the algorithmic convergence of the sample version, and give bounds to the time until perfect separation of the sample. We conclude by some results on the statistical optimality of the L_2 boosting.

Keywords: classification, Gauss-Southwell algorithm, AdaBoost, cross-validation, non-parametric convergence rate

1. Introduction

We consider a standard classification problem: Let $(X,Y),(X_1,Y_1),\ldots,(X_n,Y_n)$ be an i.i.d. sample, where $Y_i \in \{-1,1\}$ and $X_i \in \mathcal{X}$. The goal is to find a good classification rule, $\mathcal{X} \to \{-1,1\}$.

The AdaBoost algorithm was originally defined, Schapire (1990), Freund (1995), and Freund and Schapire (1996) as an algorithm to construct a good classifier by a "weighted majority vote" of simple classifiers. To be more exact, let $\mathcal{H}$ be a set of simple classifiers. The AdaBoost classifier is given by $\mathrm{sgn}\big(\sum_{m=1}^{M}\lambda_m h_m(x)\big)$, where $\lambda_m \in \mathbb{R}$, $h_m \in \mathcal{H}$, are found sequentially by the following algorithm:

0. Let $c_1 = c_2 = \cdots = c_n = 1$, and set $m = 1$.

1. Find $h_m = \arg\min_{h\in\mathcal{H}} \sum_{i=1}^{n} c_i h(X_i)Y_i$. Set

$$\lambda_m = \frac{1}{2}\log\Big(\frac{\sum_{i=1}^{n}c_i + \sum_{i=1}^{n}c_i h_m(X_i)Y_i}{\sum_{i=1}^{n}c_i - \sum_{i=1}^{n}c_i h_m(X_i)Y_i}\Big) = \frac{1}{2}\log\Big(\frac{\sum_{h_m(X_i)=Y_i}c_i}{\sum_{h_m(X_i)\neq Y_i}c_i}\Big).$$

©2006 Peter J. Bickel, Ya'acov Ritov and Alon Zakai.

```
box(1,  90.0798, 749.25,  93.1802, 744.0,   Times-Roman?, 11, #000000, 744, "J").
box(2,  93.0,    747.75,  96.9849, 744.0,   Times-Roman?, 11, #000000, 744, "o").
box(3,  97.1547, 747.75,  101.14,  743.976, Times-Roman?, 11, #000000, 744, "u").
box(4,  101.109, 747.751, 103.763, 744.0,   Times-Roman?, 11, #000000, 744, "r").
box(5,  103.852, 747.75,  107.837, 744.0,   Times-Roman?, 11, #000000, 744, "n").
box(6,  108.0,   747.75,  111.539, 744.0,   Times-Roman?, 11, #000000, 744, "a").
box(7,  111.4,   749.492, 113.615, 744.0,   Times-Roman?, 11, #000000, 744, "l").
box(8,  115.5,   747.75,  119.485, 744.0,   Times-Roman?, 11, #000000, 744, "o").
box(9,  119.597, 750.0,   122.251, 744.0,   Times-Roman?, 11, #000000, 744, "f").
box(10, 124.186, 749.25,  131.272, 744.0,   Times-Roman?, 11, #000000, 744, "M").
box(11, 131.25,  747.75,  134.789, 744.0,   Times-Roman?, 11, #000000, 744, "a").
box(12, 135.0,   747.75,  138.539, 743.999,Times-Roman?, 11, #000000, 744, "c").
box(13, 138.32,  749.491, 142.304, 744.0,   Times-Roman?, 11, #000000, 744, "h").
box(14, 142.362, 749.309, 144.578, 744.0,   Times-Roman?, 11, #000000, 744, "i").
box(15, 144.576, 747.75,  148.561, 744.0,   Times-Roman?, 11, #000000, 744, "n").
box(16, 148.634, 747.75,  152.173, 744.0,   Times-Roman?, 11, #000000, 744, "e").
box(17, 154.066, 749.25,  158.935, 744.0,   Times-Roman?, 11, #000000, 744, "L").
box(18, 159.038, 747.75,  162.577, 744.0,   Times-Roman?, 11, #000000, 744, "e").
box(19, 162.671, 747.75,  166.209, 744.0,   Times-Roman?, 11, #000000, 744, "a").
box(20, 165.953, 747.751, 168.607, 744.0,   Times-Roman?, 11, #000000, 744, "r").
box(21, 168.695, 747.75,  172.68,  744.0,   Times-Roman?, 11, #000000, 744, "n").
box(22, 172.682, 749.309, 174.898, 744.0,   Times-Roman?, 11, #000000, 744, "i").
box(23, 174.896, 747.75,  178.881, 744.0,   Times-Roman?, 11, #000000, 744, "n").
box(24, 179.194, 747.75,  183.179, 742.501,Times-Roman?, 11, #000000, 744, "g").
box(25, 184.866, 749.25,  190.182, 744.0,   Times-Roman?, 11, #000000, 744, "R").
box(26, 190.249, 747.75,  193.788, 744.0,   Times-Roman?, 11, #000000, 744, "e").
box(27, 193.5,   747.75,  196.6,   744.0,   Times-Roman?, 11, #000000, 744, "s").
box(28, 196.882, 747.75,  200.421, 744.0,   Times-Roman?, 11, #000000, 744, "e").
box(29, 200.25,  747.75,  203.789, 744.0,   Times-Roman?, 11, #000000, 744, "a").
box(30, 203.797, 747.751, 206.451, 744.0,   Times-Roman?, 11, #000000, 744, "r").
```

Fig. 3. WINE output: Vectorial Transformation of the Document

3.1 A kernel-based method to group basic blocks

The first step in the document layout analysis concerns the identification of rules to automatically shift from the basic digital document description to a higher level one. Indeed, by analyzing the PS or PDF source, the "elementary" blocks that make up the document, identified by WINE, often correspond just to fragments of words (see Figure 3), thus a first aggregation based on their overlapping or adjacency is needed in order to obtain blocks surrounding whole words (*word-blocks*). Successively, a further aggregation starting from the *word-blocks* could be performed to have blocks that group words in lines (*line-blocks*), and finally the *line-blocks* could be merged to build a paragraph (*paragraph-blocks*). As to the grouping of blocks into lines, since techniques based on the mean distance between blocks proved unable to correctly handle cases of multi-column documents, we decided to apply Machine Learning approaches in order to automatically infer rewriting rules that could suggest how to set some parameters in order to group together rectangles (words) to obtain lines. To do this, RARE (Rule Aggregation REwriter) uses a kernel-based method to learn rewriting rules able to perform the bottom-up construction of the whole document starting from the basic/word blocks up to the lines. Specifically, such a learning task was cast to a Multiple Instance Problem and solved exploiting the kernel-based algorithm proposed in [7]. In our setting, each elementary block is described by means of a feature-vector of the form:

$$[Block\_Name, Page\_No, X_i, X_f, Y_i, Y_f, C_x, C_y, H, W]$$

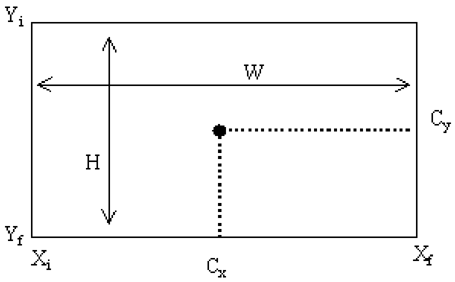


Fig. 4. Block Features

made up of parameters interpreted according to the representation in Figure 4, i.e.:

- $Block\_Name$: the identifier of the considered block;
- $Page\_No$: the number of page in which the block is positioned;
- X_i and X_f: the x coordinate values, respectively, for the start and end point of the block;
- Y_i and Y_f: the y coordinate values, respectively, for the start and end point of the block;
- C_x and C_y: the x and y coordinate values, respectively, for the centroid of the block;
- H, W: the distances (height and width) between start and end point of, respectively, x and y coordinate values.

Starting with this description of the elementary blocks, the corresponding example descriptions, from which rewriting rules have to be learned, are built considering each block along with its Close Neighbor blocks: Given a block O_n and the **C**lose **N**eighbor blocks CNO_{nk}, with their own description:
$[O_n, Page\_No, X_{ni}, X_{nf}, Y_{ni}, Y_{nf}, C_{nx}, C_{ny}, H_n, W_n]$,
$[CNO_{nk}, Page\_No, X_{nki}, X_{nkf}, Y_{nki}, Y_{nkf}, C_{nkx}, C_{nky}, H_{nk}, W_{nk}]$
we represent an example E by means of the template $[O_n, CNO_{nk}]$, i.e.:
$[New\_Block\_Name, Page\_No, X_{ni}, X_{nf}, Y_{ni}, Y_{nf}, C_{nx}, C_{ny},$
$X_{nki}, X_{nkf}, Y_{nki}, Y_{nkf}, C_{nkx}, C_{nky}, D_x, D_y]$
where the New_Block_Name is a name for the new block built by appending the names of both the original blocks, the information about the x and y coordinates are the original ones and two new parameters, D_x and D_y, contain the information about the distances between the two blocks.

Fixed a block O_n, the template $[O_n, CNO_{nk}]$ is used to find, among all the word blocks in the document, every instance of close neighbors of the considered block O_n. Such an example (set of instances) will be labelled by an expert as positive for the target concept “the two blocks can be merged” if and only if the blocks O_n and CNO_{nk} are adjacent and belong to the same line in the original document, or as negative otherwise. Figure 5 reports an example of the selected close neighbor blocks for the block $b1$. All the blocks represented with dashed lines could eventually be merged, and hence they will represent the positive instances for the concept *merge*, while dotted lines have been exploited to represent the blocks that could not be merged, and hence will represent the negative instances for the target concept. It is worth noting that not every pair of adjacent blocks has to be considered a positive example since they could belong to different frames in the considered document. Such a situation is reported in Figure 6. Indeed, typical cases in which a block is adjacent to the considered block but actually belongs to another frame are, e.g., when they belong to adjacent columns of a multi-column document (right part of Figure 6) or when they belong to two different frames of the original document (for example, the *Title* and the *Authors* frame - left part of Figure 6).

In such a representation, a block O_n has at least one close neighbor block and at most eight (CNO_{nk} with $k \in \{1, 2, \ldots, 8\}$ or, top-down, from left to right: *top_left_corner, top, top_right_corner, right, bottom_right_corner, bottom, bottom_left_corner, left*); the immediate consequence of the adopted representation is that each single example is actually made up of a bag of instances and, hence, the problem can be clearly cast as a Multiple Instance Problem to be solved by applying the Iterated-Discrim algorithm [7] in or-

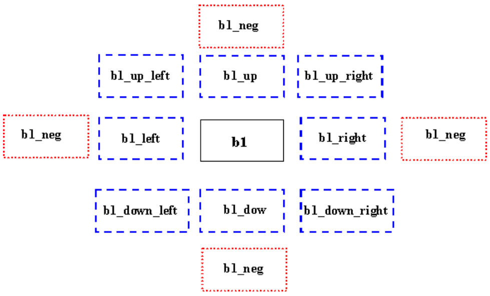


Fig. 5. Close Neighbor blocks for block $b1$

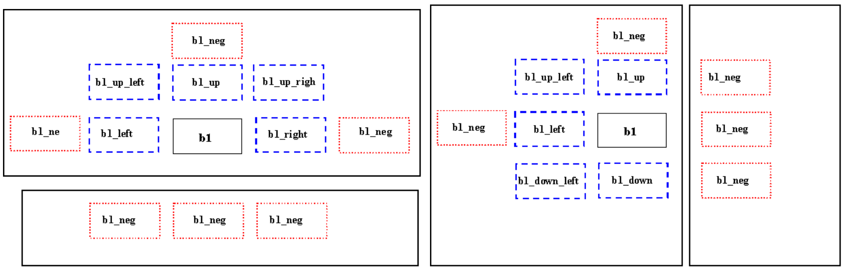


Fig. 6. Selection of positive and negative blocks according the original document: one-column document on the left, two column document on the right

der to discover the relevant features and their values to be encoded in rules made up of numerical constraints allowing to automatically set parameters to group together words in lines. In this way, the XML line-level description of the document is obtained, that represents the input to the next step in the layout analysis of the document.

In the following, an example of the representation is provided. Given the representation shown in Figure 5 for the identification of positive and negative blocks, and the template for the example description, a possible representation for the positive example (a set of instances) expressing the description “block b35 can be merged with blocks b36,b34, b24, b43 if and only if such blocks have the reported numeric features (size and position in the document)” is:

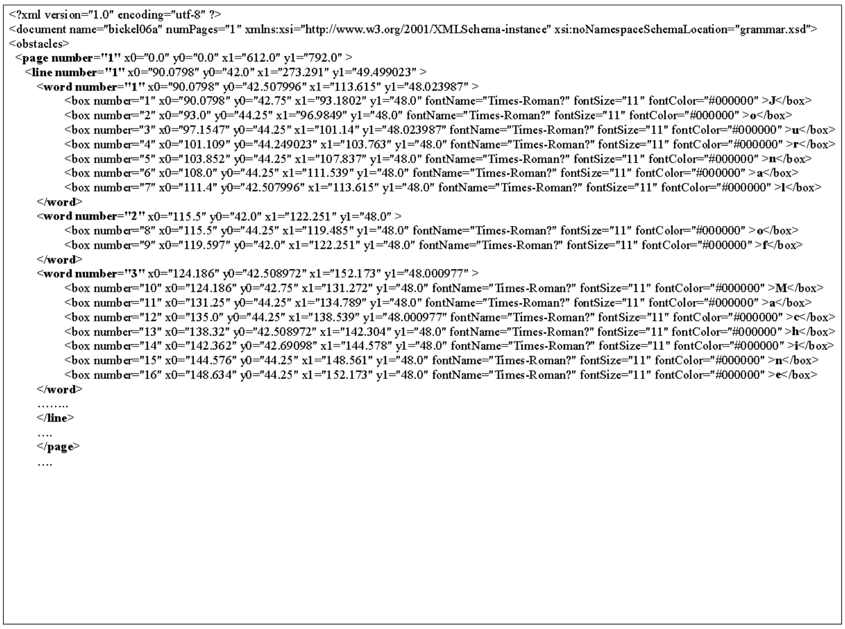
```
ex(b35) :-
  istance([b35, b36, 542.8, 548.3, 447.4, 463.3, 553.7, 594.7,
              447.4, 463.3, 545.6, 455.3, 574.2, 455.3, 5.5, 0]).
  istance([b35, b34, 542.8, 548.3, 447.4, 463.3, 529.2, 537.4,
              447.4, 463.3, 545.5, 455.4, 533.3, 455.3, 5.5, 0]).
  istance([b35, b24, 542.8, 548.3, 447.4, 463.3, 496.3, 583.7,
              427.9, 443.8, 545.5, 455.3, 540.1, 435.9, 0, 3.5]).
  istance([b35, b43, 542.8, 548.3, 447.4, 463.3, 538.5, 605.4,
              466.9, 482.8, 545.5, 455.3, 571.9, 474.8, 0, 3.5]).
```

3.2 Discovery of the background structure of the document

The objects that make up a document are spatially organized in *frames*, defined as collections of objects completely surrounded by white space. It is worth noting that 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 frames, while a paragraph might begin in one column and continue into the next column.

The next step towards the discovery of the document logical structure, after transforming the original digital document into its basic XML representation and grouping the basic blocks into lines, consists in performing the layout analysis of the document by applying an algorithm named DOC (Document Organization Composer), a variant of that reported in [8] for addressing the key problem in geometric layout analysis. DOC analyzes the whitespace and background structure of each page in the document in terms of rectangular covers, and it is efficient and easy to implement.

Once DOC has identified the whitespace structure of the document, thus yielding the background, it is possible to compute its complement, thus obtaining the document content blocks. When computing the complement, two levels of description are generated. The former refers to single blocks filled with the same kind of content, the latter consists in rectangular frames that



```
<?xml version="1.0" encoding="utf-8" ?>
<document name="bickel06a" numPages="1" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="grammar.xsd">
<obstacles>
 <page number="1" x0="0.0" y0="0.0" x1="612.0" y1="792.0" >
  <line number="1" x0="90.0798" y0="42.0" x1="273.291" y1="49.499023" >
   <word number="1" x0="90.0798" y0="42.507996" x1="113.615" y1="48.023987" >
        <box number="1" x0="90.0798" y0="42.75" x1="93.1802" y1="48.0" fontName="Times-Roman?" fontSize="11" fontColor="#000000" >J</box>
        <box number="2" x0="93.0" y0="44.25" x1="96.9849" y1="48.0" fontName="Times-Roman?" fontSize="11" fontColor="#000000" >o</box>
        <box number="3" x0="97.1547" y0="44.25" x1="101.14" y1="48.023987" fontName="Times-Roman?" fontSize="11" fontColor="#000000" >u</box>
        <box number="4" x0="101.109" y0="44.249023" x1="103.763" y1="48.0" fontName="Times-Roman?" fontSize="11" fontColor="#000000" >r</box>
        <box number="5" x0="103.852" y0="44.25" x1="107.837" y1="48.0" fontName="Times-Roman?" fontSize="11" fontColor="#000000" >n</box>
        <box number="6" x0="108.0" y0="44.25" x1="111.539" y1="48.0" fontName="Times-Roman?" fontSize="11" fontColor="#000000" >a</box>
        <box number="7" x0="111.4" y0="42.507996" x1="113.615" y1="48.0" fontName="Times-Roman?" fontSize="11" fontColor="#000000" >l</box>
   </word>
   <word number="2" x0="115.5" y0="42.0" x1="122.251" y1="48.0" >
        <box number="8" x0="115.5" y0="44.25" x1="119.485" y1="48.0" fontName="Times-Roman?" fontSize="11" fontColor="#000000" >o</box>
        <box number="9" x0="119.597" y0="42.0" x1="122.251" y1="48.0" fontName="Times-Roman?" fontSize="11" fontColor="#000000" >f</box>
   </word>
   <word number="3" x0="124.186" y0="42.508972" x1="152.173" y1="48.000977" >
        <box number="10" x0="124.186" y0="42.75" x1="131.272" y1="48.0" fontName="Times-Roman?" fontSize="11" fontColor="#000000" >M</box>
        <box number="11" x0="131.25" y0="44.25" x1="134.789" y1="48.0" fontName="Times-Roman?" fontSize="11" fontColor="#000000" >a</box>
        <box number="12" x0="135.0" y0="44.25" x1="138.539" y1="48.000977" fontName="Times-Roman?" fontSize="11" fontColor="#000000" >c</box>
        <box number="13" x0="138.32" y0="42.508972" x1="142.304" y1="48.0" fontName="Times-Roman?" fontSize="11" fontColor="#000000" >h</box>
        <box number="14" x0="142.362" y0="42.69098" x1="144.578" y1="48.0" fontName="Times-Roman?" fontSize="11" fontColor="#000000" >i</box>
        <box number="15" x0="144.576" y0="44.25" x1="148.561" y1="48.0" fontName="Times-Roman?" fontSize="11" fontColor="#000000" >n</box>
        <box number="16" x0="148.634" y0="44.25" x1="152.173" y1="48.0" fontName="Times-Roman?" fontSize="11" fontColor="#000000" >e</box>
   </word>
   ……..
   </line>
   ….
   </page>
   ….
```

Fig. 7. DOC output: XML Representation of the Layout Structure of the Document of Figure 3

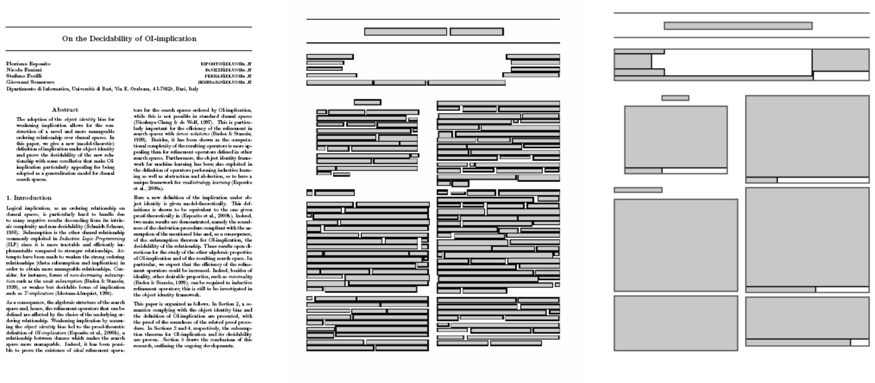
may be made up of many blocks of the former type. Thus, the overall description of the document includes both kinds of objects, plus information on which frames include which blocks and on the actual spatial relations between frames and between blocks in the same frame (e.g., above, touches, etc.). This allows to maintain both levels of abstraction independently. Figure 7 reports the *XML layout structure* that is the output of DOC. Figure 8 depicts, along with the original document, the graphical representation of the XML generated by a two-column document trough the basic block vectorial transformation and the grouped words/lines representation, obtained by means of a process that is not a merely syntactic transformation from PS/PDF to XML.

It is worth to note that exploiting as-is the algorithm reported in [8] on the basic representation discovered by the WINE tool in real document domains turns out to be unfeasible due to the usually large number of basic blocks discovered. Thus, the preliminary aggregation of basic blocks into words and then of words into lines by means of the above procedure is fundamental for the efficiency and effectiveness of the DOC algorithm. Additionally, some modifications to the algorithm on which DOC is based deserve attention. First of all, any horizontal/vertical line in the layout is considered as a natural separator, and hence is already considered as background (along with all the surrounding white space) before the algorithm starts. Second, any white block whose height or width is below a given threshold is discarded as insignificant (this should avoid returning inter-word or inter-line spaces). Lastly, since the original algorithm tries to find iteratively the maximal white rectangles, taking it to its natural end and then computing the complement would result again in the original basic blocks coming from the previous steps and provided as input. This would be useless, and hence raised the problem of identifying a stop criterion to end this process.

Such a criterion was empirically established as the moment in which the area of the new white rectangle retrieved, $W(R)$, represents a percentage of the total white area in the document page, $W(D)$, less than a given threshold δ, i.e.:
Let $A(D)$ be the area of the document page under consideration, $A(R_i),\ i = 1, \ldots, n$ be the areas of the blocks identified thus far in the page, and $W(D) = A(D) - \sum_{i=1,\ldots,n} A(R_i)$ be the total white area in the page (computed as the difference between the total page area and the area of the blocks in the page), then the stop criterion is established as:

$$\frac{W(R)}{W(D)} < \delta$$

The empirical study was performed applying the algorithm in full on a set of 100 documents of three different categories, and it took into account the values of three variables in each step of the algorithm: number of new white rectangles (black line in Figure 9) normalized between 0 and 1, ratio of the last white area retrieved with respect to the total white area of the current page of the document (bold line in Figure 9), ratio of the white area retrieved so far with respect to the total white area of the current page of the document (dashed line in Figure 9). The ratio of the white area retrieved, the dashed line, is never equal to 1 (the algorithm does not find all the white area), but it becomes stable before reaching 1/4 of the total steps of the algorithm. Such a consideration is generally valid for all the documents except for those having



On the Decidability of OI-implication

Floriana Esposito ESPOSITO@DI.UNIBA.IT
Nicola Fanizzi FANIZZI@DI.UNIBA.IT
Stefano Ferilli FERILLI@DI.UNIBA.IT
Giovanni Semeraro SEMERARO@DI.UNIBA.IT
Dipartimento di Informatica, Università di Bari, Via E. Orabona, 4 I-70126, Bari, Italy

Abstract

The adoption of the *object identity* bias for weakening implication allows for the construction of a novel and more manageable ordering relationship over clausal spaces. In this paper, we give a new (model-theoretic) definition of implication under object identity and prove the decidability of the new relationship with some corollaries that make OI-implication particularly appealing for being adopted as a generalization model for clausal search spaces.

1. Introduction

Logical implication, as an ordering relationship on clausal spaces, is particularly hard to handle due to many negative results descending from its intrinsic complexity and non-decidability (Schmidt-Schauss, 1988). Subsumption is the other clausal relationship commonly exploited in *Inductive Logic Programming* (ILP) since it is more tractable and efficiently implementable compared to stronger relationships. Attempts have been made to weaken the strong ordering relationships (theta subsumption and implication) in order to obtain more manageable relationships. Consider, for instance, forms of *non-decreasing subsumption* such as the *weak subsumption* (Badea & Stanciu, 1999), or weaker but decidable forms of implication such as *T-implication* (Idestam-Almquist, 1996).

As a consequence, the algebraic structure of the search space and, hence, the refinement operators that can be defined are affected by the choice of the underlying ordering relationship. Weakening implication by assuming the *object identity* bias led to the proof-theoretic definition of *OI-implication* (Esposito et al., 2000b), a relationship between clauses which makes the search space more manageable. Indeed, it has been possible to prove the existence of *ideal* refinement operators for the search spaces ordered by OI-implication, while this is not possible in standard clausal spaces (Nienhuys-Cheng & de Wolf, 1997). This is particularly important for the efficiency of the refinement in search spaces with *dense solutions* (Badea & Stanciu, 1999). Besides, it has been shown as the computational complexity of the resulting operators is more appealing than for refinement operators defined in other search spaces. Furthermore, the object identity framework for machine learning has been also exploited in the definition of operators performing inductive learning as well as abstraction and abduction, so to have a unique framework for *multistrategy learning* (Esposito et al., 2000a).

Here a new definition of the implication under object identity is given model-theoretically. This definitions is shown to be equivalent to the one given proof-theoretically in (Esposito et al., 2000b). Indeed, two main results are demonstrated, namely the soundness of the derivation procedure compliant with the assumption of the mentioned bias and, as a consequence, of the subsumption theorem for OI-implication, the decidability of the relationship. These results open directions for the study of the other algebraic properties of OI-implication and of the resulting search space. In particular, we expect that the efficiency of the refinement operators could be increased. Indeed, besides of ideality, other desirable properties, such as *minimality* (Badea & Stanciu, 1999), can be required to inductive refinement operators; this is still to be investigated in the object identity framework.

This paper is organized as follows. In Section 2, a semantics complying with the object identity bias and the definition of OI-implication are presented, with the proof of the soundness of the related proof procedure. In Sections 3 and 4, respectively, the subsumption theorem for OI-implication and its decidability are proven. Section 5 draws the conclusions of this research, outlining the ongoing developments.

Fig. 8. Line and final layout analysis representations of the generated XML structure of a document

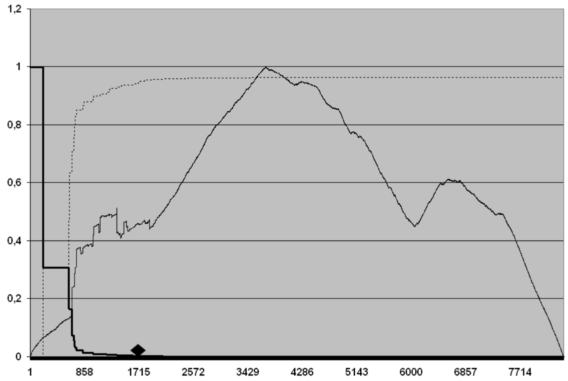


Fig. 9. Stop Criterion Analysis

a scattered appearance. Such a point, highlighted in the figure with a black diamond, is the best stop point for the algorithm since before it the layout is not sufficiently detailed, while after it useless white spaces are found, as shown with the black line in the graphic. Indeed, this is the point in which all the useful white spaces in the document, e.g. those between columns and sections, have been identified. Such a consideration is confirmed by analyzing the trend of the ratio of the last white area retrieved with respect to the total white area in the current page of the document (bold line), that decreases up to 0 in such a point. This suggests to stop executing the algorithm just there. It is worth noting that this value is reached very early, and before the size of the structure containing the blocks waiting to be processed starts growing dramatically, thus saving lots of time and space resources.

4 Structural Metadata Extraction

The organization of the document collection and the extraction of the interesting text is a fundamental issue for a more efficient storage and retrieval process in a digital library. To perform such tasks, one has to firstly identify the correct type the document belongs to (e.g. understand whether the document is a magazine, or a book, or a scientific paper) in order to file it in the corresponding record. Then, the significant components of the document have to be identified in order to extract from them the information needed to categorize it. Since carrying out manually such a process is unfeasible due to the huge amount of documents, our proposal is the use of a concept learning system to infer rules able to correctly classify the document type along with its significant components. The inborn complexity of the document domain, and the need to express relations among components, suggests the exploitation of symbolic first-order logic as a powerful representation language to handle such a situation. Furthermore, based on the belief that in typical digital libraries on the Internet new documents continuously become available over time and are to be integrated in the collection, we consider incrementality as a fundamental requirement for the techniques to be adopted. Even more difficult, it could be the case that not only single definitions turn out to be faulty and need revision, but whole new document classes are to be included in the collection as soon as the first document for them become available. This represents a problem for most existing systems, that require not only all the information on the application domain to be available when the learning process starts, but also the set of classes for which they must learn definitions to be completely defined since the beginning.

These considerations, among others about the learning systems available in the literature, led to the exploitation of INTHELEX (INcremental THEory Learner from EXamples) [12], whose most characterizing features are its incremental nature, the reduced need of a deep background knowledge, the exploitation of negative information and the peculiar bias on the generalization model, which reduces the search space and does not limit the expressive power of the adopted representation language.

4.1 The Learning System

INTHELEX is an Inductive Logic Programming [13] system that learns hierarchical logic theories from positive and negative examples. It is fully incremental (in addition to the possibility of refining previously generated hypotheses/definitions, learning can also start from an empty theory), and adopts Datalog^{OI} [14] as a representation language: based on the Object Identity assumption (different symbols must denote different objects), it ensures effectiveness of the descriptions and efficiency of their handling, while preserving the expressive power of the unrestricted case. It can learn simultaneously multiple concepts/classes, possibly related to each other; it can retain all the processed examples, so to guarantee validity of the learned theories on all of them.

INTHELEX has a closed loop architecture (i.e., feedback on performance is used to activate the theory revision phase [15]). The learning cycle it performs, depicted in Figure 10, can be described as follows. A set of examples of the concepts to be learned, possibly selected by an expert, is provided by the environment. This set can be subdivided into three subsets (training, tuning, and test set) according to the way in which examples are exploited during the learning process. Specifically, training examples, previously classified by the expert, are stored in the base of processed examples, and exploited to obtain an initial theory that is able to explain them. In INTHELEX, such a theory can also be provided by the expert, or even be empty. Subsequently, the validity of the theory against new available tuning/test examples, also stored in the example base as long as they are processed, is checked against the set of inductive hypotheses, producing a decision that is compared to the correct

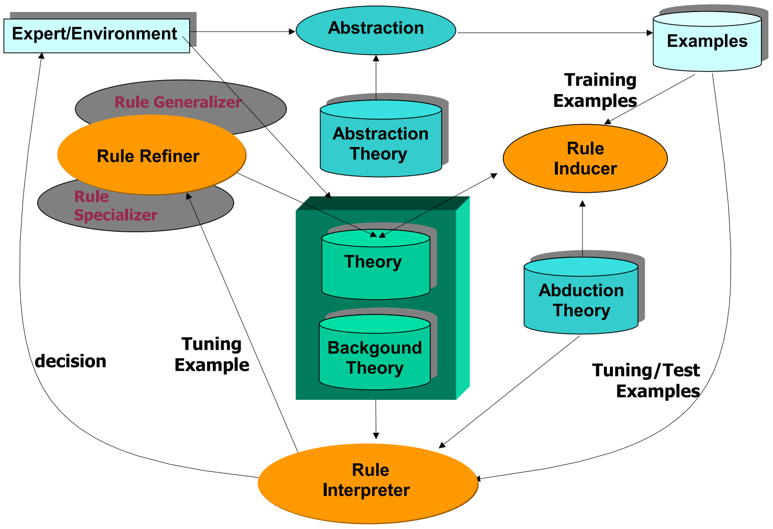


Fig. 10. Learning System Architecture

one. Test examples are exploited just to check the predictive capabilities of the theory, intended as its behavior on new observations, without causing a refinement of the theory in the case of incorrectness. Conversely, in case of incorrectness on a tuning example, the cause of the wrong decision can be located and the proper kind of correction chosen, firing the theory revision process. In this way, tuning examples are exploited incrementally to modify incorrect theories according to a data-driven strategy.

Specifically, INTHELEX incorporates two inductive refinement operators to revise the theory, one for generalizing definitions that reject positive examples, and the other for specializing definitions that explain negative examples. If an example is positive and not covered, the system first tries to generalize one of the available definitions of the concept the example refers to, so that the resulting revised theory covers the new example and is consistent with all the past negative examples. If such a generalization is found, then it replaces the chosen definition in the theory, or else a new clause is chosen to compute generalization. If no definition can be generalized in a consistent way, the system checks whether the example itself can represent a new alternative (consistent) definition of the concept. If so, such a definition is added to the theory, or else the example itself is added as an exception. If the example is negative and covered, specialization is needed. Among the theory definitions that concur in covering the example, INTHELEX tries to specialize one by adding to it one or more conditions which characterize all the past positive examples and can discriminate them from the current negative one. In case of failure, the system tries to add the negation of a condition, that is able to discriminate the negative example from all the past positive ones. If this fails too, the negative example is added to the theory as an exception. New incoming observations are always checked against the exceptions before applying the rules that define the concept they refer to.

Another peculiarity in INTHELEX is the embedding of multistrategy operators that may help in solving the theory revision problem by pre-processing the incoming information. It was operated according to the theoretical framework for integrating different learning strategies known as Inferential Theory of Learning [16]. Deduction refers to the possibility of better representing the examples and, consequently, the inferred theories. INTHELEX exploits deduction to recognize known concepts that are implicit in the examples description and explicitly add them to the descriptions. The system can be provided with a *Background Knowledge*, supposed to be correct and hence not modifiable, containing (complete or partial) concept definitions to be exploited during deduction. Differently from abstraction (see next), all the specific information used by deduction is left in the example description. Hence, it is preserved in the learning process until other evidence reveals it is not significant for the concept definition, which is a more cautious behavior. Abduction was defined by Peirce as hypothesizing facts that, together with a given theory, could explain a given observation, and aims at completing possibly partial information in the examples (adding more details). According to the framework proposed in [17], this can be done by exploiting a set of *abducibles* (concepts about which assumptions can be made, that carry all the incompleteness of the domain: if it were possible to complete their definitions then the theory would be correctly described) and a set of *integrity constraints* (each corresponding to a combination of conditions that is not allowed to occur, that provide indirect information about abducibles). Abstraction is a pervasive activity in human perception and reasoning, and aims at removing superfluous details from the description of both the examples and the theory. Thus, the exploitation of abstraction results in the shift from the language in which the theory is described to a higher level one. According to the framework proposed in [18], in INTHELEX abstraction takes place by means of a set of operators that replace a number of components by a compound object, or decrease the granularity of a set of values, or ignore whole objects or just part of their features, or neglect the number of occurrences of some kind of object.

4.2 Representation Language

In order to work, the learning system must be provided with a suitable first-order logic representation of the documents. Thus, once the layout components of a document are automatically discovered as explained in Section 3, the next step concerns the automatic description of the pages, blocks and frames according to their size, spatial [19] and inclusion relations. Dealing with multi-page documents, the document description must be enriched with *page information* such as: page number and position (whether it is at the beginning, in the middle or at the end of the document, and specifically whether it is the

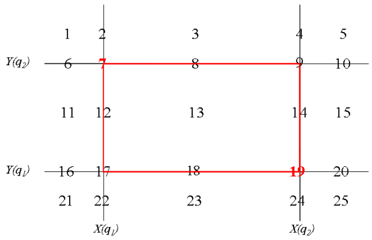


Fig. 11. Representation Plans according to [19]

last one), total number of pages in the document. As pointed out, the automatic process ends with a set of content rectangles recognized in each single page. Such rectangles are described by means of their size (height and width), 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 frames and pages, e.g. $contain(page_i, frame_j)$, and between blocks and frames, e.g. $contain(frame_j, block_k)$.

Another possible relation between rectangles is the spatial one. Given a rectangle r, one can ideally divide the plan containing it in 25 parts (see Figure 11), and describe the relative position between the other rectangles and r in terms of the plans they occupy with respect to r. Such a technique is applied to every block belonging to a same frame and to all the adjacent frames, where a rectangle is adjacent to another rectangle r if it is the nearest rectangle to r in some plan. Additionally, such a kind of representation of the plans allows also to express in the example description the topological relations [20, 19], such as closeness, intersection and overlapping between rectangles. However, the topological information can be deduced by the spatial relationships, and thus it can be included by the system during the learning process by means of deduction and abstraction. For instance, the following fragment of background knowledge could be provided to the system to infer the topological relations between two blocks or frames:

```
top_alignment(B1,B2):-
        occupy_plane_9(B1,B2), not(occupy_plane_4(B1,B2)).
top_alignment(B1,B2):-
       occupy_plane_10(B1,B2), not(occupy_plane_5(B1,B2)).
bottom_alignment(B1, B2) :-
      occupy_plane_19(B1, B2), not(occupy_plane_24(B1, B2)).
bottom_alignment(B1, B2) :-
      occupy_plane_20(B1, B2), not(occupy_plane_25(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)).
right_alignment(B1, B2) :-
      occupy_plane_19(B1, B2), not(occupy_plane_20(B1, B2)).
```

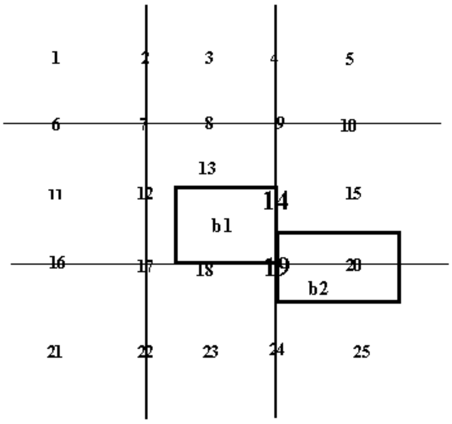


Fig. 12. Block representation

```
right_alignment(B1, B2) :-
      occupy_plane_24(B1, B2), not(occupy_plane_25(B1, B2)).
touch(B1,B2):-
       occupy_plane_14(B1,B2), not(occupy_plane_13(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)).
touch(B1,B2):-
       occupy_plane_19(B1,B2), not(occupy_plane_13(B1,B2)).
```

Thus, given the representation of the two blocks reported in Figure 12 where block B2 occupies plans 14, 15, 19, 24, 25 while block B1 occupies plans 13, 14, 18, 19, and having in common the plans 14 and 19, the initial representation will be made up, among other descriptors, by:

```
......, occupy_plane_14(b2, b1), occupy_plane_19(b2, b1), ......
```

and the system is able to recognize the topological relations above reported giving the following:

```
..., touch(b2,b1), bottom_alignment(b2,b1),....
```

In this language unary predicate symbols, called attributes, are used to describe properties of a single layout component (e.g. height and length), while n-ary predicate symbols, called relations, are used to express spatial relationships between layout components. A complete list of attributes and relations is reported in Table 1.

4.3 Layout Correction

Due to the fixed stop threshold (see section 3.2), it might happen that after the layout analysis step some blocks are not correctly recognized, i.e. background

Page Descriptors

page_number(d,p): p is the number of current page in document d
last_page(p): true if page p is the last page of the document
in_first_pages(p): true if page p belongs to the first n pages of the document (n < 1/3 total number of the pages in the document)
in_middle_pages(p): true if page p is in the middle n pages of the document (1/3 < n < 2/3 total number of the pages in the document)
in_last_pages_pagine(p): true if page p belongs to the last n pages of the document (n > 2/3 total number of the pages in the document)
number_of_pages(d, n): n is the total number of pages in document d
page_width(p,w): w is the page width (a value normalized in [0,1])
page_height(p,h): h is the page height (a value normalized in [0,1])

Frame/Block Descriptors

frame(p,f): f is a frame of page p
block(p,b): b is a block of page p
type(b,t): t is the type of the block content (text, graphic, mixed, empty, vertical_line, horizontal_line, oblique_line)
width(b,w): w is the block width in pixels
height(b,h): h is the block height in pixels
x_coordinate_rectangle(r,x): x is the horizontal coordinate of the start point of the rectangle (frame or block) r
y_coordinate_rectangle(r,y): y is the vertical coordinate of the start point of the rectangle (frame or block) r

Topological Relation Descriptors

belong(b, f): block b belongs to frame f
pos_upper(p, r): rectangle r is positioned in the upper part of page p
pos_middle(p, r): the rectangle r is positioned in the middle part of page p (in vertical sense)
pos_lower(p, r): the rectangle r is positioned in the lower part of page p
pos_left(p, r): the rectangle r is positioned in the left part of page p
pos_center(p, r): the rectangle r is positioned in the center part of page p (in horizontal sense)
pos_right(p, r): the rectangle r is positioned in the right part of page p
touch(b1,b2): block b1 touches block b2 and vice versa
on_top(b1,b2): block b1 is positioned on block b2
to_right(b1,b2): block b1 is positioned on the right of block b2
top_alignment(b1, b2): block b1 is over block b2
bottom_alignment(b1, b2): block b1 is under block b2
left_alignment(b1, b2): block b1 is on the left of block b2
right_alignment(b1, b2): block b1 is on the right of block b2

Table 1. Attributes/Relations used to describe the documents

areas are considered as content ones and/or vice versa. In such a case a phase of layout correction would be desirable. A first correction of the automatically recognized layout can be performed by allowing the system user to manually force further forward steps, or to go some step backward, in the algorithm with respect to the stop threshold. This is possible since the system maintains three structures that keep track of: all white rectangles found (W), all black rectangles found (B) and all rectangles that it has not analyzed yet (N: if no threshold is given all the rectangles are analyzed and N will be empty at the end of processing). Hence, when the user is not satisfied by the discovered layout because some background is missing, he can decide to go forward, and the system will extract and process further rectangles from N. Conversely, if the user notes that the system has found insignificant background pieces, he can decide to go back and the system will correspondingly move blocks between W, B and N.

However, such a solution is not always effective in case of lost significant background rectangles (e.g., small areas that represent the cut point between two frames), since they could be very small and hence it would be necessary to perform many forward steps (during which the system would probably restore insignificant white rectangles) before being able to retrieve them. Even worse, the system could be completely unable to retrieve the needed background just because it is too small to satisfy the constraints.

To solve both problems, DOCG (Document Organization Correction Generator), a module to improve the analysis performed by DOC, was implemented. It uses machine learning techniques to automatically infer rules for recognizing interesting background rectangles among those discarded or not yet analyzed by the layout analysis algorithm, according to their size and position with respect to the surrounding blocks. Specifically, we first processed a number of documents, then presented to the user all the blocks in the N structure and asked him to force as background some rectangles that the system had erroneously discarded (even if such rectangles do not satisfy the constraints), and to remove insignificant rectangles erroneously considered as background by the system. These blocks were then considered as examples for the learning system in order to infer rules to automatically perform this task during future layout analysis processes. Again, due to the need of expressing many relationships among blocks in order to represent these situations, a first-order description language was required, and INTHELEX was exploited as a learning system. Specifically, each example described the block to be forced plus all the blocks around it, along with their size and position in the document, both before and after the manual correction.

Classification

After detecting the document layout structure, a logic role can be associated to some of its components. In fact, the role played by a layout component represents meta-information that could be exploited to tag the document and help its filing and management within the digital library. The logical components can be arranged in another hierarchical structure, which is called logical structure. The logical structure is the result of repeatedly dividing the content of a document into increasingly smaller parts, on the basis of the human-perceptible meaning of the content. The leaves of the logical structure are the basic logical components, such as authors and title of a magazine article or the date and signature in a commercial letter. The heading of an article encompasses the title and the author and is therefore an example of composite logical component. Composite logical components are internal nodes of the logical structure. The root of the logical structure is the document class. The problem of finding the logical structure of a document can be cast as the problem of associating some layout components with a corresponding logical component.

The first component that can be tagged is the document itself, according to the class it belongs to (*document image classification* step). Indeed, in general many kinds of documents with different layout structures can be present in one library, and they have to be exploited in different ways according to their type. In turn, the type of a document is typically reflected by the layout structure of its first page: e.g., humans can immediately distinguish a bill from an advertisement or a letter or a (newspaper or scientific) article without actually reading their content, but just based on their visual appearance.

For this reason, we decided to apply machine learning to infer rules that allow to automatically classify new incoming documents according to their first-page layout, in order to determine how to file them in the digital repository and what kind of processing they should undergo next. This step turns out to be very significant in a digital library, where a lot of different layout structures for the documents, either belonging to different classes or even to the same class, can be encountered. Again, the diverse and complex relationships that hold between the layout components of a document suggested the use of a first-order representation language and learning system. Additionally, the possibility of continuously extending the repository with new classes of documents or with modifications of the existing ones asked for incremental abilities that INTHELEX provides.

Classification of multi-page documents is performed by matching the layout structure of their first page against the automatically learned models of classes of documents. These models capture the invariant properties of the images layout structures of documents belonging to the same class. They consist of rules expressed in a first-order logic language, so that the document classification problem can be reformulated as a matching test between a logic formula that represents a model and another logic formula that describes the image/layout properties of the first page. The choice of a first-order logic language fulfils the requirements of flexibility and generality.

Understanding

Once the class of a document has been identified on the basis of its first page layout, its logical components that are present in any page can be located and tagged by matching the layout structure of each page against models of logical components. Indeed, if we assume that it is possible to identify logical components by using only layout information, just as humans do, these models capture the invariant layout properties of the logical components of documents in the same class.

This is the task of the *document image understanding* phase, that must necessarily follow document image classification since the kind of logical components that can be expected in a document strongly depends on the document class (e.g., in a commercial letter one expects to find a sender, possibly a logo, an address, an object, a body, a date and a signature, whereas in a scientific paper one could be interested in its title, authors and their affiliations, abstract and bibliographic references). Once again, they are expressed as rules in a first-order logic language. However, differently from document classification, the document understanding problem cannot be effectively reformulated as a simple matching test between logic formulae. The association of the logical description of pages with logical components requires a full-fledged theorem prover, since it is typical that one component is defined and identified in relation to another one.

5 Categorization, Filing and Indexing

One of the most important tasks in digital library management concerns the categorization of documents. Effectiveness in performing such a task represents the success factor in the retrieval process, in order to identify documents that are really interesting for the users. Indeed, a problem of most existing word-based retrieval systems consists in their ineffectiveness in finding interesting documents when the users exploit different words than those by which the information they seek has been indexed. This is due to a number of tricky features that are typical of natural language: different writers use different words to describe the same idea (*synonymy*), thus a person issuing a query in a search engine might use different words than those that appear in an interesting document, and could not retrieve the document; one word can have multiple meanings (*polysemy*), so a searcher can get uninteresting documents concerning the alternate meanings. To face such a problem, the Latent Semantic Indexing (LSI) technique [21] has been adopted, that tries to overcome the weaknesses of term-matching based retrieval by treating the unreliability of observed term-document association data as a statistical problem. Indeed, LSI assumes that there exists some underlying latent semantic structure in the data, that is partially obscured by the randomness of word choice with respect to the retrieval phase, and that can be estimated by means of statistical techniques.

As a weighting function, a combination of the local and global relevance of terms has been adopted in the following way:

$$w_{ij} = L(i,j) * G(i)$$

where $L(i,j)$ represents the local relevance of the term i in the document j and $G(i)$ represents the global value of the term i. A good way to relate such values is represented by the log entropy function, where:

$$L(i,j) = \log(tf_{ij} + 1)$$

$$G(i) = 1 - \sum_{j=1,\dots,N} \frac{p_{ij} * \log(p_{ij})}{\log(N)}$$

here, N represents the number of documents and $p_{ij} = \frac{tf_{ij}}{gf_i}$, where tf_{ij} is the local relevance for each term (the frequency of the term i in the document j, TF) and gf_i is the global relevance for each term (the frequency of the term i in the whole set of documents, IDF). This way, the logarithmic value of the local factor $L(i,j)$ mitigates the effects due to large variations in term frequencies, while the entropy function of the global factor $G(i)$ mitigates the noise that could be present in the documents.

The success of the retrieval step results strictly related to the choice of the parameter k that represents the best new rank, lower than the original one, to reduce the matrix. In our system, it is set as the minimum number of the documents needed to cover the whole set of terms. As to the retrieval phase, the following weighting function was applied to each element of the query vector in order to create, for the query too, the correspondence between the local and global factor:

$$q_{ij} = (0.5 + \frac{0.5 * tf_i}{maxtf}) * \log \frac{N}{n}$$

where tf_i is the frequency of term i in the query, $maxtf$ is the maximum value among all the frequencies, N represents the total number of documents and n is the number of documents in which term i appears. In our system the *cosine similarity* function [22] was exploited to perform the comparison between the query vector and each document vector. Documents that show a high degree of similarity according to the value computed are those interesting for the user query.

However, the large amount of items that a document management system has to deal with, and the continuous flow of new documents that could be added to the initial database, require an incremental methodology to update the initial LSI matrix. Indeed, applying from scratch at each update the LSI method, taking into account both the old (already analyzed) and the new documents, would become computationally inefficient. Two techniques have been developed in the literature to update (i.e., add new terms and/or documents to) an existing LSI generated database: Folding-In [23] and SVD-Updating [24]. An analysis on the performance of both techniques shows that SVD-Updating is more suitable to be exploited in a digital library environment. Indeed, the former is a much simpler alternative that uses the existing SVD to represent new information but yields poor-quality updated matrices, since the semantic correlation information contained in the new documents/terms is not exploited by the updated semantic space. The latter represents a trade-off between the former and the recomputation from scratch.

6 Exploitation and Evaluation

The system for automated digital documents processing was evaluated in each step, from document acquisition to document indexing for categorization and information retrieval purposes. Since the system can be embedded as a document management engine into many different domain-specific applications, in this section we focus on the Conference Management scenario. As we will see DOMINUS can usefully support some of the more critical and knowledge-intensive tasks involved by the organization of a scientific conference, such as the assignment of the submitted papers to suitable reviewers.

6.1 Scientific Conference Management Scenario

Organizing scientific conferences is a complex and multi-faceted activity that often requires the use of a Web-based management system to make some tasks a little easier to carry out, such as the job of reviewing papers. Some of the features typically provided by these packages are: submission of abstracts and papers by Authors; submission of reviews by the Program Committee Members (PCMs); download of papers by the Program Committee (PC); handling of reviewers preferences and bidding; Web-based assignment of papers to PCMs for review; review progress tracking; Web-based PC meeting; notification of acceptance/rejection; sending e-mails for notifications.

Let us now present a possible scenario. An Author connects to the Internet and opens the submission page, where (after registering, or after logging in if already registered) he can browse his hard disk and submit a paper by choosing the corresponding file in one of the accepted formats. After uploading, the paper undergoes the following processing steps. The layout analysis algorithm is applied, in order to single out its layout components. Then, it is translated into a first-order logic description and classified by a proper module according to the theory learned so far for the acceptable submission layout standards. A single conference can allow different layout standards for the submitted papers (e.g., full paper, poster, demo) and it can be the case that many conferences have to be managed at the same time. Depending on the identified class, a further step consists in locating and labelling the layout components of interest for that class (e.g., title, author, abstract and references in a full paper).

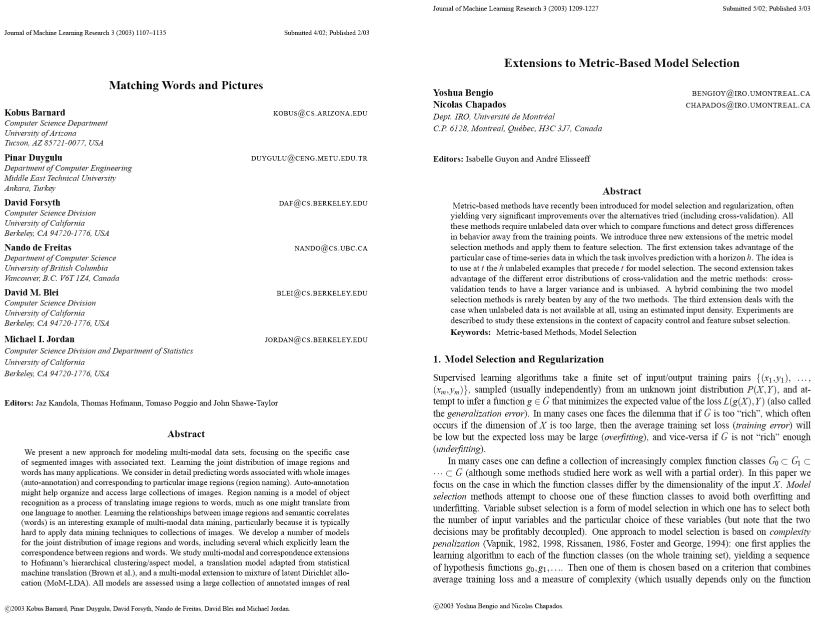
The text contained in each of such components is read, stored and used to automatically file the submission record (e.g., by filling its title, authors and abstract fields). If the system is unable to carry out any of these steps, such an event is notified to the Conference administrators, that can manually fix the problem and let the system complete its task. Such manual corrections are logged and used by the incremental learning component to refine the available classification/labeling theories in order to improve their performance on future submissions. Nevertheless, this is done off-line, and the updated theory replaces the old one only after the learning step is successfully completed, thus allowing further submissions in the meantime. Alternatively, the corrections can be logged and exploited all at once to refine the theory when the system performance falls below a given threshold.

The next step, which is currently under investigation, concerns the automatic categorization of the paper content on the grounds of the text it contains. This allows to match the paper topic against the reviewers' expertise, in order to find the best associations for the final assignment. Specifically, we exploit the text in the title, abstract and bibliographic references, assuming that they concentrate the subject and research field the paper is concerned with. This requires a pre-processing step that extracts the meaningful content from each reference (ignoring, e.g., page numbers, place and editors). Furthermore, the paper topics discovered in the indexing phase are matched with the conference topics with the aim of supporting the conference scheduling.

6.2 Experimental results

In the above scenario, the first step concerns document image classification and understanding of the documents submitted by the Authors. In order to evaluate the system on this phase, experiments were carried out on a fragment of 353 documents coming from our digital library, made up of documents of the last ten years available in online repositories (i.e., publishers' online sites, authors' home pages, the Scientific Literature Digital Library CiteSeer, our submissions, etc.) interesting for our research topics. The resulting dataset is made up of four classes of documents: the Springer-Verlag Lecture Notes in Computer Science (LNCS) series, the Elsevier journals style (ELSEVIER), the Machine Learning Journal (MLJ) and the Journal of Machine Learning Research (JMLR). Specifically, 70 papers were formatted according to the LNCS style (proofs and initial submission of the papers), 61 according to the ELSEVIER style, 122 according to the MLJ (editorials, Kluwer Academy and Springer Science publishers) style and 100 according to the JMLR style.

It is worth to note that even documents in the same class might fall in different layout standard, according to the period of publication, since the publisher layout style may have changed in time. Thus, the changes in spatial organization of the first page might affect the classification step (see Figure 13).



Journal of Machine Learning Research 3 (2003) 1107–1135 Submitted 4/02; Published 2/03

Matching Words and Pictures

Kobus Barnard KOBUS@CS.ARIZONA.EDU
Computer Science Department
University of Arizona
Tucson, AZ 85721-0077, USA

Pinar Duygulu DUYGULU@CENG.METU.EDU.TR
Department of Computer Engineering
Middle East Technical University
Ankara, Turkey

David Forsyth DAF@CS.BERKELEY.EDU
Computer Science Division
University of California
Berkeley, CA 94720-1776, USA

Nando de Freitas NANDO@CS.UBC.CA
Department of Computer Science
University of British Columbia
Vancouver, B.C. V6T 1Z4, Canada

David M. Blei BLEI@CS.BERKELEY.EDU
Computer Science Division
University of California
Berkeley, CA 94720-1776, USA

Michael I. Jordan JORDAN@CS.BERKELEY.EDU
Computer Science Division and Department of Statistics
University of California
Berkeley, CA 94720-1776, USA

Editors: Jaz Kandola, Thomas Hofmann, Tomaso Poggio and John Shawe-Taylor

Abstract

We present a new approach for modeling multi-modal data sets, focusing on the specific case of segmented images with associated text. Learning the joint distribution of image regions and words has many applications. We consider in detail predicting words associated with whole images (auto-annotation) and corresponding to particular image regions (region naming). Auto-annotation might help organize and access large collections of images. Region naming is a model of object recognition as a process of translating image regions to words, much as one might translate from one language to another. Learning the relationships between image regions and semantic correlates (words) is an interesting example of multi-modal data mining, particularly because it is typically hard to apply data mining techniques to collections of images. We develop a number of models for the joint distribution of image regions and words, including several which explicitly learn the correspondence between regions and words. We study multi-modal and correspondence extensions to Hofmann's hierarchical clustering/aspect model, a translation model adapted from statistical machine translation (Brown et al.), and a multi-modal extension to mixture of latent Dirichlet allocation (MoM-LDA). All models are assessed using a large collection of annotated images of real

©2003 Kobus Barnard, Pinar Duygulu, David Forsyth, Nando de Freitas, David Blei and Michael Jordan.

Journal of Machine Learning Research 3 (2003) 1209-1227 Submitted 5/02; Published 3/03

Extensions to Metric-Based Model Selection

Yoshua Bengio BENGIOY@IRO.UMONTREAL.CA
Nicolas Chapados CHAPADOS@IRO.UMONTREAL.CA
Dept. IRO, Université de Montréal
C.P. 6128, Montreal, Québec, H3C 3J7, Canada

Editors: Isabelle Guyon and André Elisseeff

Abstract

Metric-based methods have recently been introduced for model selection and regularization, often yielding very significant improvements over the alternatives tried (including cross-validation). All these methods require unlabeled data over which to compare functions and detect gross differences in behavior away from the training points. We introduce three new extensions of the metric model selection methods and apply them to feature selection. The first extension takes advantage of the particular case of time-series data in which the task involves prediction with a horizon h. The idea is to use at t the h unlabeled examples that precede t for model selection. The second extension takes advantage of the different error distributions of cross-validation and the metric methods: cross-validation tends to have a larger variance and is unbiased. A hybrid combining the two model selection methods is rarely beaten by any of the two methods. The third extension deals with the case when unlabeled data is not available at all, using an estimated input density. Experiments are described to study these extensions in the context of capacity control and feature subset selection.
Keywords: Metric-based Methods, Model Selection

1. Model Selection and Regularization

Supervised learning algorithms take a finite set of input/output training pairs $\{(x_1,y_1), \ldots, (x_m,y_m)\}$, sampled (usually independently) from an unknown joint distribution $P(X,Y)$, and attempt to infer a function $g \in G$ that minimizes the expected value of the loss $L(g(X),Y)$ (also called the *generalization error*). In many cases one faces the dilemma that if G is too "rich", which often occurs if the dimension of X is too large, then the average training set loss (*training error*) will be low but the expected loss may be large (*overfitting*), and vice-versa if G is not "rich" enough (*underfitting*).

In many cases one can define a collection of increasingly complex function classes $G_0 \subset G_1 \subset \cdots \subset G$ (although some methods studied here work as well with a partial order). In this paper we focus on the case in which the function classes differ by the dimensionality of the input X. *Model selection* methods attempt to choose one of these function classes to avoid both overfitting and underfitting. Variable subset selection is a form of model selection in which one has to select both the number of input variables and the particular choice of these variables (but note that the two decisions may be profitably decoupled). One approach to model selection is based on *complexity penalization* (Vapnik, 1982, 1998, Rissanen, 1986, Foster and George, 1994): one first applies the learning algorithm to each of the function classes (on the whole training set), yielding a sequence of hypothesis functions $g_0, g_1, \ldots$. Then one of them is chosen based on a criterion that combines average training loss and a measure of complexity (which usually depends only on the function

©2003 Yoshua Bengio and Nicolas Chapados.

Fig. 13. Two first pages from the JMLR class

This calls for the incremental abilities of the incremental system that must generate different concept definitions at the same time. Indeed, the system is able, at any moment, to learn the layout description of a new class of document style preserving the correct definition of the others. In this way a global theory is build, containing the definitions of different document styles, that could be used for many conferences.

Each document was described according to the features reported in Section 4.2, and was considered as 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 the training and test sets contained the same percentage of positive and negative examples. Furthermore, the system was provided with background knowledge expressing topological relations (see Section 4.2), and abstraction operators were used to discretize numeric values concerning size and position into intervals expressed by symbolic descriptors. In the following, an example of the abstraction rules for rectangles width discretization is given.

```
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.
```

```
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.
```

A first experiment was run to infer the document classification rules; good results were obtained in terms of runtime, predictive accuracy, number of theory revisions (Rev = total revisions, RevPos = revisions performed on positive examples only, RevNeg = revisions performed on negative examples). Furthermore, in order to evaluate the theory revision rate, some additional measures were considered: the global percentage of revisions Rev on the whole training set (RevRate), the percentage of revisions RevPos on the positive examples (RevRatePos) and the percentage of revisions RevNeg on the negative examples (RevRateNeg)), as reported in Table 2. The lowest accuracy and poorest performance was obtained on MLJ, that reflects the variety of corresponding paper formats and typing styles.

Table 2. Learning System Performance: inferring rules for class paper identification

Class	Rev	RevPos	RevNeg	RevRate	RevRatePos	RevRateNeg	Time (s.)	Acc. %
LNCS	16	11.7	4.3	0.05	0.18	0.02	662.88	97.2
MLJ	28.2	18.7	9.5	0.08	0.17	0.04	2974.87	93.5
ELSEVIER	13.6	11.2	2.4	0.04	0.20	0.01	303.85	98.9
JMLR	12.7	10	2.7	0.04	0.11	0.01	1961.66	98.2

As to the revision rate, Figure 14 sketches the system performance with respect to revisions and accuracy on the training phase in the classification step in one fold (the nearest to the average reported in Table 2). The curve represents the trend in accuracy as long as new examples are analyzed, while the cuts represent the revision points. These points become very sparse as the number of analyzed examples increases and the accuracy curve, after a first phase in which many revisions have to be performed to restore the theory correctness, tends to increase towards a stable condition. The results concerning class MLJ are perfectly consistent with the composition of the selected sample; the variety of typing conventions and formats underlying the documents requires to extend the training set.

Once the classification step has been completed the image document understanding phase starts. The second experiment was performed on the *title*, *authors*, *abstract* and *references* layout components of documents belonging to the LNCS class. This class was chosen since it represents the layout standard of papers submitted to the 18th Conference on Industrial & Engineering Applications of Artificial Intelligence & Expert Systems (IEA/AIE 2005) which has been used as a real testbed. In Table 3 the averaged results of the 10 folds

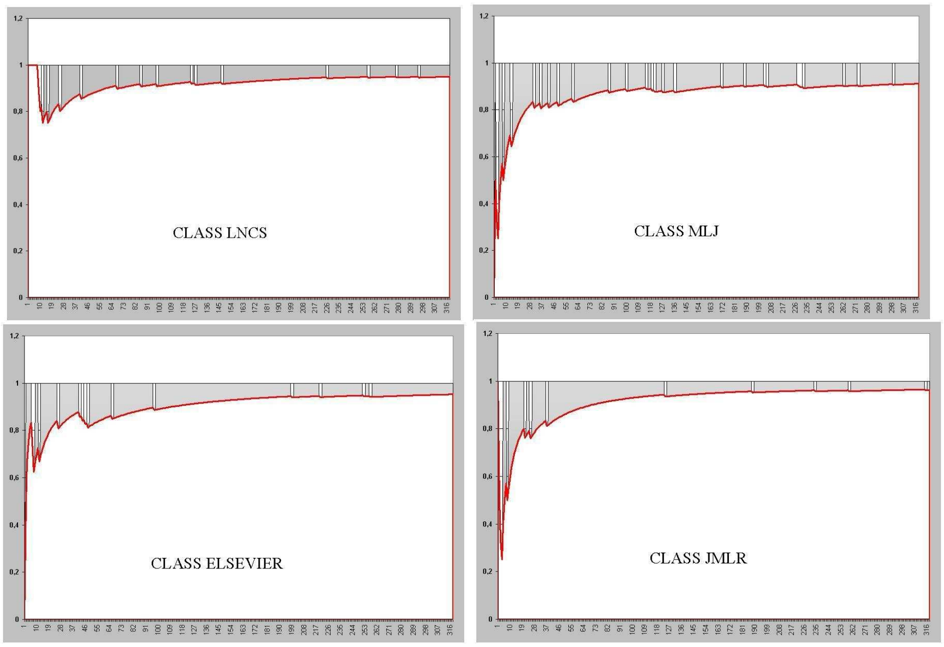


Fig. 14. Accuracy and revision rate of the learning system on tuning phase

are reported, that can be considering satisfying from both the accuracy and the time consuming point of view.

Table 3. Learning System Performance: inferring rules for components label identification

Label	Rev	RevPos	RevNeg	RevRate	RevRatePos	RevRateNeg	Time (s.)	Acc. %
Title	16.5	13.7	2.8	0.06	0.22	0.01	217.60	95.3
Abstract	10.5	9.4	1.1	0.04	0.15	0.01	104.07	96.2
Author	14.6	11.1	3.5	0.05	0.17	0.02	146.48	98.6
Ref	15.4	10.6	4.8	0.06	0.17	0.02	150.93	97.4

A very hard task in the organization of Scientific Conferences is the reviewers assignment; due to the many constraints, manually performing such a task is very tedious and difficult, and does not guarantee the best results. The proposed document management system can assist the conference program chair both in indexing and retrieving the documents and their associated topics, although not explicitly reported by the paper authors. In the following we present an experiment carried out on the above reported dataset consisting of 264 papers submitted to the IEA/AIE 2005 conference, whose Call for Papers included 34 topics of interest.

Firstly, the layout of each paper in digital format was automatically analyzed in order to recognize the significant components. In particular, the abstract and title were considered the most representative of the document subject, and hence the corresponding text was extracted to apply the LSI technique. The words contained therein were stemmed according to the technique proposed by Porter [25], resulting in a total of 2832 word stems. Then, the same procedure was applied to index the reviewers expertise according to the titles of their papers appearing in the DBLP Computer Science Bibliography repository (http://www.informatik.uni-trier.de/∼ley/db/), resulting in 2204 stems.

In both cases, the LSI parameters were set in such a way that all the conference topics were covered as different concepts. The experiment consisted first in performing 34 queries, each corresponding to one conference topic, both on papers and on reviewers, and then in associating respectively to each paper/reviewer the first l results of the LSI queries. The results obtained on document topic recognition showed that considering 88 documents per query is enough to cover the whole set of documents. However, considering just 30 documents per query, 257 out of 264 documents (97.3%) were already assigned to at least one topic. This is an acceptable trade-off since the remaining 7 documents can be easily assigned by hand. Moreover, 30 documents are a good choice to assure the equidistribution over the document. Interestingly, more than half of the documents (54.7%) concern $2 \div 4$ topics so confirming the extremely specialized nature of the conference and the high correlation between the topics. The results, compared to the conference program chair indications, showed a 79% accuracy on average. Setting $l = 10$, the automatic assignment of the topics to the reviewers resulted in 65% accuracy compared to the suggestions of the conference program chair.

Lastly, the expert system GRAPE (Global Review Assignment Processing Engine) [26] has the task of automatically assigning the papers to reviewers taking into account specific knowledge (i.e., conflicts, nationality, common interest, etc.). The final assignments were considered very useful suggestions by the experts so confirming the goodness of the indexing process and of the topic associations.

7 Related Work

Image Document analysis refers to algorithms and techniques developed in order to obtain a computer-readable description of a scanned document [27].

While an impressive amount of contribution has been presented applied to scanned image documents, only recently a few works have faced the problem of handling digital document formats such as PDF and PS. Most of them aim at extracting (some part of) the document content by means of a syntactic parsing of the PDF [28, 29, 30] or at discovering the background by means of statistical analyses applied to the numerical features of the documents and its components [31]. A further step towards digital document analysis as opposed (but complementary) to document image analysis is represented by the work reported in [32]; here, a method is proposed for the extraction of the logical structure from PDF files by examining the visual appearance and geometric position of text and image blocks distributed over the whole document and exploiting the information on line spacing and font usage in order to bridge the semantic gap between the document image and its content. In this work, the PDF file is firstly decomposed by syntactically parsing it, then grouping words into lines (by means of APIs provided by the Acrobat Exchange viewer), lines in bins (based on their point size, font name and their coordinates in the page) and finally bins in blocks. Successively, relationships (greater/lesser status) among two blocks are discovered by analyzing their features (font name and point size) and labels of the discovered blocks are identified by applying (and possibly modifying after new blocks are evaluated) a set of rules purposely codified by a domain expert for the class/tags at hand.

Recently, some works [33, 34] proposed a strategy that mixes the layout extraction methods from digital documents with the most widely used document analysis techniques. The approach consists into three steps:

- parsing syntactically the PDF file to extract the document primitives (text, image or graphics);
- recognizing and grouping homogeneous entities among the extracted primitives;
- extracting the logical layout structure by means of text entities labelling (e.g., title, author, body) and document modelling in which the entities are *projected* in a document model.

Here, for each class of documents an expert provides a model, representing its grammar, i.e. a hierarchy of logical entities.

A similar approach which uses grammars to annotate document components is proposed in [35]. Here, based on the model provided by an expert, a set of possible roles is assigned to each layout object. Then, they are collected into more complex objects until the logical structure is produced.

All the approaches reported above perform geometric layout analysis by means of a syntactic parsing of the document. Then, the mapping between the geometric and logical structure is supported by using a template of the document, a grammar representing its layout, or an expert system whose knowledge base must be provided by a human expert.

8 Conclusion

The huge amount of documents available in digital form and the flourishing of digital repositories raise problems concerning document management, concerned with effectiveness and efficiency of their successive retrieval, that cannot be faced by manual techniques. This paper proposed DOMINUS, an intelligent system characterized by the intensive application of Machine Learning techniques as a support to all phases of automated document processing, from document acquisition to document understanding and indexing. The core of DOMINUS is the Learning Server, a suite of different inductive learning methods and systems, among which the more suitable for the specific document processing phase is chosen and applied. The most interesting is INTHELEX, a proprietary incremental learning system able to handle structural descriptions and to automatically revise first-order theories.

Experiments in the real-world domain of automatic Scientific Conference Management have been presented and discussed, showing the validity of the proposed approach.

Different future work directions are planned for the proposed system. First of all, the automatic processing of bibliographic references, that can improve the identification of the document subject and context. Secondly, the use of ontologies in text processing in order to improve the effectiveness of content-based retrieval.

References

1. Esposito, F., Malerba, D., Semeraro, G., Ferilli, S., Altamura, O., Basile, T.M.A., Berardi, M., Ceci, M., Mauro, N.D.: Machine learning methods for automatically processing historical documents: From paper acquisition to XML transformation. In: Proceedings of the First International Workshop on Document Image Analysis for Libraries (DIAL 2004). (2004) 328–335
2. Berners Lee, T., Hendler, J., Lassila, O.: The semantic web. Scientific American **284**(5) (2001) 34–43
3. Utgoff, P.E.: Incremental induction of decision trees. Machine Learning **4**(2) (1989) 161–186
4. Cauwenberghs, G., Poggio, T.: Incremental and decremental support vector machine learning. In: Advances in Neural Information Processing Systems (NIPS 2000). Volume 13., Cambridge, MA, USA, MIT Press (2000) 409–415
5. Solomonoff, R.: Progress in incremental machine learning. In: NIPS Workshop on Universal Learning Algorithms and Optimal Search, Dec. 14, 2002, Whistler, B.C., Canada, 27 pp. (2003)
6. Wong, W., Fu, A.: Incremental document clustering for web page classification. In: IEEE 2000 Int. Conf. on Info. Society in the 21st century: emerging technologies and new challenges (IS2000), Nov 5-8, 2000, Japan. (2000)
7. Dietterich, T.G., Lathrop, R.H., Lozano-Perez, T.: Solving the multiple instance problem with axis-parallel rectangles. Artificial Intelligence **89**(1-2) (1997) 31–71
8. Breuel, T.M.: Two geometric algorithms for layout analysis. In: Workshop on Document Analysis Systems. (2002)
9. Glunz, W.: (pstoedit - a tool converting postscript and PDF files into various vector graphic formats) (http://www.pstoedit.net).
10. Adobe Systems Inc.: PostScript language reference manual – 2nd ed. Addison Wesley (1990)
11. Adobe Systems Inc.: PDF Reference version 1.3 – 2nd ed. Addison Wesley (2000)
12. Esposito, F., Ferilli, S., Fanizzi, N., Basile, T.M., Di Mauro, N.: Incremental multistrategy learning for document processing. Applied Artificial Intelligence: An Internationa Journal **17**(8/9) (2003) 859–883
13. Muggleton, S., Raedt, L.D.: Inductive logic programming: Theory and methods. Journal of Logic Programming **19/20** (1994) 629–679
14. Semeraro, G., Esposito, F., Malerba, D., Fanizzi, N., Ferilli, S.: A logic framework for the incremental inductive synthesis of datalog theories. In Fuchs, N., ed.: Proceedings of the 7th International Workshop on Logic Program Synthesis and Transformation. Volume 1463 of LNCS., Springer (1998) 300–321
15. Becker, J.: Inductive learning of decision rules with exceptions: Methodology and experimentation. Master's thesis, Dept. of Computer Science, University of Illinois at Urbana-Champaign, Urbana, Illinois (1985) B.S. diss., UIUCDCS-F-85-945.
16. Michalski, R.: Inferential theory of learning. developing foundations for multistrategy learning. In Michalski, R., Tecuci, G., eds.: Machine Learning. A Multistrategy Approach. Volume IV. Morgan Kaufmann (1994) 3–61
17. Kakas, A., Mancarella, P.: On the relation of truth maintenance and abduction. In: Proceedings of the 1st Pacific Rim International Conference on Artificial Intelligence, Nagoya, Japan (1990)

18. Zucker, J.D.: Semantic abstraction for concept representation and learning. In Michalski, R.S., Saitta, L., eds.: Proceedings of the 4th International Workshop on Multistrategy Learning. (1998) 157–164
19. Papadias, D., Theodoridis, Y.: Spatial relations, minimum bounding rectangles, and spatial data structures. International Journal of Geographical Information Science **11**(2) (1997) 111–138
20. Egenhofer, M.: Reasoning about binary topological relations. In Gunther, O., Schek, H.J., eds.: Second Symposium on Large Spatial Databases. Volume 525 of Lecture Notes in Computer Science., Springer (1991) 143–160
21. Deerwester, S.C., Dumais, S.T., Landauer, T.K., Furnas, G.W., Harshman, R.A.: Indexing by Latent Semantic Analysis. Journal of the American Society of Information Science **41**(6) (1990) 391–407
22. Baeza-Yates, R.A., Ribeiro-Neto, B.A.: Modern Information Retrieval. ACM Press / Addison-Wesley (1999)
23. Berry, M.W., Dumais, S.T., O'Brien, G.W.: Using linear algebra for intelligent information retrieval. SIAM Rev. **37**(4) (1995) 573–595
24. O'Brien, G.W.: Information management tools for updating an SVD-encoded indexing scheme. Technical Report UT-CS-94-258, University of Tennessee (1994)
25. Porter, M.F.: An algorithm for suffix stripping. In Karen, J.S., Willet, P., eds.: Readings in information retrieval. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1997) 313–316
26. Di Mauro, N., Basile, T.M.A., Ferilli, S.: GRAPE: An expert review assignment component for scientific conference management systems. In: Innovations in Applied Artificial Intelligence: 18th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems (IEA/AIE 2005). Volume 3533 of Lecture Notes in Computer Science., Springer Verlag (2005) 789–798
27. Nagy, G.: Twenty years of document image analysis in PAMI. IEEE Transactions on Pattern Analysis and Machine Intelligence **22**(1) (2000) 38–62
28. Futrelle, R.P., Shao, M., Cieslik, C., Grimes, A.E.: Extraction, layout analysis and classification of diagrams in PDF documents. In: Proceedings of Seventh International Conference on Document Analysis and Recognition (ICDAR 2003). (2003) 1007–1014
29. Chao, H.: Graphics extraction in PDF document. In Kanungo, T., Smith, E.H.B., Hu, J., Kantor, P.B., eds.: Proceedings of SPIE - The International Society for Optical Engineering. Volume 5010. (2003) 317–325
30. Ramel, J.Y., Crucianu, M., Vincent, N., Faure, C.: Detection, extraction and representation of tables. In: Proceedings of the Seventh International Conference on Document Analysis and Recognition (ICDAR 2003), Washington, DC, USA, IEEE Computer Society (2003) 374–378
31. Chao, H., Fan, J.: Layout and content extraction for pdf documents. In: Document Analysis Systems VI, Proceeding of the Sixth International Workshop (DAS 2004). Volume 3163 of Lecture Notes in Computer Science., Springer Verlag (2004) 213–224
32. Lovegrove, W.S., Brailsford, D.F.: Document analysis of PDF files: methods, results and implications. Electronic Publishing – Origination, Dissemination and Design **8**(2-3) (1995) 207–220
33. Hadjar, K., Rigamonti, M., Lalanne, D., Ingold, R.: Xed: A new tool for extracting hidden structures from electronic documents. In: DIAL '04: Proceedings of

the First International Workshop on Document Image Analysis for Libraries (DIAL'04), Washington, DC, USA, IEEE Computer Society (2004) 212

34. Rigamonti, M., Bloechle, J.L., Hadjar, K., Lalanne, D., Ingold, R.: Towards a canonical and structured representation of PDF documents through reverse engineering. In: ICDAR '05: Proceedings of the Eighth International Conference on Document Analysis and Recognition, Washington, DC, USA, IEEE Computer Society (2005) 1050–1055
35. Anjewierden, A.: AIDAS: Incremental logical structure discovery in pdf documents. In: Proceedings of Sixth International Conference on Document Analysis and Recognition (ICDAR 2001). (2001) 374–378