Automatic Content-based Indexing of Digital Documents through Intelligent Processing Techniques

Floriana Esposito Stefano Ferilli Teresa M.A. Basile Nicola Di Mauro
Dipartimento di Informatica
University of Bari
Via Orabona, 4 - Bari, Italy
E-mail: {esposito,ferilli,basile,ndm}@di.uniba.it

Abstract

The availability of large, heterogeneous repositories of electronic documents is increasing rapidly, and the need for flexible, sophisticated document manipulation tools is growing correspondingly. This work presents DOMINUS a system for automated electronic documents processing characterized by the intensive exploitation of intelligent techniques in each step of the process from document acquisition in electronic format to document indexing for categorization and information retrieval purposes. It embeds incremental learning techniques useful in the context of web libraries where documents are acquired in an incremental fashion. Since the system is general and flexible, it can be embedded as a document management engine into many different domain-specific applications. Here, we present the exploitation of the system on the Conference Management domain, and show how DOMINUS can usefully support some of the more critical and knowledge-intensive tasks involved by the organization of a scientific conference.

1. Introduction and Motivation

Knowledge distribution and preservation in time has always been a fundamental issue in the history of cultural development. Since knowledge can be conveyed by many different kinds of documents, document management assumed a key role in this perspective. Since several centuries up to a few decades ago, the only support available for dumping and spreading information was paper, which caused nearly the totality of our legacy material to be in the form of printed paper documents, often in few copies stored in archives and libraries. This obviously represents a serious obstacle to wide access and distribution of the information content they bear, which caused the need for document processing techniques aimed at ensuring preservation and access of the available material.

A significant support to such a concern has been provided thanks to the power and flexibility offered by automatic techniques developed by computer science. The current spread of computer systems throughout all human activities has been in the last years a source for both the need and the opportunity that a significant amount of documents are available in electronic form. Indeed, having the documents in the form of computer files, thanks to the Internet, makes it far easier and cheaper (in terms of time and money) their sending, transportation, exchange, storage and retrieval. Specifically, in the last decades, the great diffusion of computers on one side, and of the Internet on the other, caused a migration or duplication of many documents from the paper support to the digital one. This greatly facilitates copy, distribution and accessibility of the documents themselves, and hence clearly fulfils the preservation requirements. As a consequence, a huge amount of documents in digital format are spread throughout the World Wide Web in the most diverse websites, and a whole research area focused on principles and techniques for setting up and managing document collections in the form of Digital Libraries has started and quickly developed. The result of such a situation is the development of Digital Libraries, intended as distributed collections of textual or multimedia documents, whose main concerns and problems consist in the acquisition and organization of the information contained therein.

However, this is in itself not sufficient to solve the whole problem of content-based access: on the contrary, it also poses new challenges to the computer science researchers, and raises a number of problems for their effective handling, which in turn require the development of different approaches and techniques that are able to deal with their peculiarities and characteristics. More precisely, the question is, given collections of digitized documents, how is it possible to discover among them useful knowledge to be used as meta-information to support their retrieval and management. This encompasses several sub-problems, involving issues that range from the analysis of the document layout, to the identification of the document type and of the role played by its components, up to the exploitation of techniques for document content indexing. The obvious solution of manually creating and maintaining an updated index (in the form of a database of extracted information) is clearly infeasible, due to the potentially huge amount of data to be handled, tagged and indexed. This causes a strong motivation for the research concerned with methods that can provide solutions for automatically acquiring new knowledge.

An easy solution for the legacy material has been digitization. As a side-effect, this has caused the need to digitize a huge amount of documents in paper format, which has typically been obtained through image scanning, possibly followed by *tout-court* application of Optical Character Recognition techniques. However, such an approach turns out to be computationally quite expensive, and is not of use for identifying specific interesting parts of the document, and hence great amounts of research have been devoted to improve the efficiency and effectiveness of such a process, in order to go beyond simple image representation, exploiting the different components in different ways and tagging them according to their semantic role [22]. Also our research group devoted in the past great effort towards the application of machine learning techniques as a support for the automatic management of paper documents. Such techniques were successfully applied to different domains, among which cultural heritage material [15].

More recently, the situation has significantly changed, although paper has not lost its predominance as a means for conveying information. Indeed, the perspective has radically changed as well: documents are no more simply digitized images of an original paper counterpart, but are composed and produced directly in electronic form, of which the paper printout is just a facility that improves physical handling and readability. Thus, almost all documents nowadays are generated natively in digital format by means of word processors, text editors and similar, thus obtaining their paper-printed counterpart only as a side-effect that can be delayed until the actual need of such a physical support because of law or practical requirements (which often does not happen at all).

Here, differently from the past techniques that work on scanned image documents, we focus on electronic documents in PostScript (PS) or Portable Document Format (PDF), that represent the current de facto standard for document interchange and hence cover the greatest part of the available material. The specific problem we will deal with in the following consists in the application of intelligent techniques to a system for the management of a collection of electronic documents (specifically, we will focus on the domain of scientific papers) on the Internet, aimed at automatically extracting significant information from the documents themselves in order to organize them in a way that can be useful to properly store, retrieve and manage them in a Semantic Web [6] perspective. Indeed, organizing the documents on the grounds of the knowledge they contain is fundamental for being able to correctly access them according to particular needs in various situations. In particular, in our application domain, 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 and 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 hence it requires the ability to handle multi-page documents, and because it in turn is made up of different sub-components carrying different kinds of information, to be handled and exploited in different ways.

Three processing stages are typically needed to identify a document significant components. Layout Analysis consists in the perceptual organization process that aims at identifying the single blocks that make up a document and at detecting relations among them, expressed by the so-called layout structure. Hence, the association of the proper logical role to each of such components of the document yields the so-called logical structure. Since the logical structure is obviously different according to the kind of document, two steps are in charge of identifying such a structure. Document Image Classification aims at associating the document to a class that expresses its type (e.g., scientific paper). Document Image Understanding aims at identifying the significant layout components for that class and at associating to each of them a tag that expresses its role (e.g., title, author, abstract, etc.). A large amount of knowledge is required to this purpose [3, 11] and in the literature a great effort is made to hand-code such a knowledge [22, 18, 27]. We propose to exploit Machine Learning techniques to carry out these two steps on electronic documents. In particular, the need for expressing relations among layout components requires the use of symbolic first-order techniques. Indeed, the classical attribute-value language is not sufficiently expressive to describe the relations. For example, to formalize the sentence “A is the title of the book B” we have to use an atom: title_book_B = A. But now let us consider the sentence “C is the title of the book D”. Again, it must be formalized as another atom and the obvious similarities between these two sentences (they both mention the same relation) cannot be expressed. Hence, we need a more express language that satisfies the following requirements: It should be able to distinguish between objects (such as A and C) and relations about objects. Secondly, the same object should be denoted everywhere by the same symbol (formalizing the sentence “A is a title” and “A is a part of a document”, the two formalization should both contain the same symbol denoting A). In the same way the relation “is the title” should be denoted by the same symbol everywhere. Finally, we would like to use variables denoting different objects. These requirements are satisfied by the first-order language.

Furthermore, the continuous flow of new documents, which is a typical feature in Digital Libraries, calls for *incremental* abilities that can revise a faulty knowledge previously acquired for identifying the logical structure of a document. Traditional application of machine learning to intelligent systems development involves collecting a set of training examples, expressed by means of a representation language in a representation space that facilitates learning, and using a learning algorithm to induce a set of concepts from the codified training examples. These induced concepts are subsequently validated and incorporated into an inferential system and deployed into an environment. However, classical approaches to learning task require that all the information needed for the task is available at the beginning of the induction step. Learning systems that use such an approach are known as *batch* or *one-step* systems. Hence, the batch approach at learning is clearly not suitable for the knowledge assimilation process that requires the ability to incrementally revise a domain theory as new data are encountered. Indeed, real-world applications, and Digital Libraries in particular, require autonomous or semi-autonomous operation and adaptation to changes in the domain, the environment, or the user. If any of these factors changes, the classical approach requires that the entire theory development process is restarted to produce a model capable of coping with the new scenario. Such requirements suggest that incremental learning, as opposed to batch one, is needed whenever either incomplete information is available at the time of initial theory generation, or the nature of the concepts evolves dynamically. Indeed, incremental processing allows for continuous responsiveness to the environment, can potentially improve efficiency and has the ability to deal with concept evolution. Obviously, the incremental setting implicitly assumes that the information (observations) gained at any given moment is incomplete, and thus that any learned theory is (could be) potentially susceptible of changes. This, in turn, prevents the possibility of taking unchangeable decisions and definitively cutting away, regarding them as useless, portions of the search space that seem to be outside the correct (target) concept definition region at a given time.

This work presents the current version of the prototype DOMINUS (DOcument Management INtelligent Universal System), a system for automated electronic documents processing characterized by the intensive exploitation of intelligent techniques in each step of the process from document acquisition to document indexing for categorization and information retrieval purposes. Since the system is general and flexible, it can be embedded as a document management engine into many different domain-specific applications. Here, we focus on the Conference Management domain, and show how 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.

The paper is organized as follows. The next Section presents the layout analysis step performed by DOMINUS on electronic documents along with the improvements implemented in the system by means of the kernel-based approaches. Then, the incremental first-order techniques for the layout correction, classification and understanding of the document, aiming at the meta data extraction step, are analyzed. Successively, the latent semantic analysis technique for the efficient document indexing and retrieval is presented along with its peculiarity. Finally, we present the evaluation of the system DOMINUS on a real world application domain, the conference management one.

2. Layout Structure Recognition

Any document can be progressively partitioned into a hierarchy of abstract representations, called its *layout structure*. The leaves of the corresponding layout tree (lowest level of the abstraction hierarchy) are the basic layout components, such as words or images, while the root represents the whole document. In multi-page documents, the root represents a set of pages. A page may group together several layout components, called frames, which are rectangular areas of interest in the document page. The perceptual organization process that aims at detecting structures among blocks is called the *layout analysis*. An ideal layout analysis should produce a set of frames, each of which can be associated with a distinct logical component, such as title and author of a scientific paper.

Here we describe an approach implemented for discovering a full layout hierarchy in digital documents available in PostScript (PS) or Portable Document Format (PDF), based primarily on layout information. The PostScript [1] language is a simple interpretative programming language with powerful graphical capabilities that allow it to precisely describe any page. The PDF [2] language is an evolution of PostScript that is rapidly gaining acceptance as a promising file format for electronic documents. Like PostScript, it is an open standard, enabling integrated solutions from a broad range of vendors, which is the reason why we initially focused on these two languages.

Our layout analysis process starts with a preliminary preprocessing step based on a method, named `WINE` (Wrapper for the Interpretation of Non-uniform Electronic document formats), that takes as input a PDF or a PS document and produces a corresponding description of the document in vector format, expressed in XML. A similar algorithm is `PSTOEDIT` [16], that translates PostScript and

PDF graphics into other vector formats. In particular, `WINE` is a rewriting of basic PostScript operators that turns the instructions into objects. For example, the PostScript instruction to display a text becomes an object describing a text with attributes for the geometry (location on the page) and appearance (font, color, etc.). `WINE` represents the first step that transforms a PostScript document into its corresponding *XML basic representation* describing the initial digital document as a set of pages, each of which is composed of *basic blocks*. Here we present the descriptors used by `WINE` for the document representation:

1. **box**(id,x0,y0,x1,y1,font,dimension,RGB,row,string) - The text made up the document is represented by the bounding box that contains its;
2. **stroke**(id,x0,y0,x1,y1,RGB,thickness) - The graphical (horizontal/vertical) lines of the document;
3. **fill**(id,x0,y0,x1,y1,RGB) - Closed areas filling with one color;
4. **image**(id,x0,y0,x1,y1) - Raster images;
5. **page**(n,w,h) - Page information;

where: id is the identifier of the block; (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 line); font is the the type font; dimension represents the dimension of the text; RGB is the color of the text, of the line or of the 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 is the width of the page; h is the height of the page.

2.1. A kernel-based method to group basic blocks

The first step in the document layout analysis concerns the identification of rules to automatically shift an electronic document description from the basic one to an higher level description. Indeed, by analyzing the PS or PDF source, it is possible to identify the "elementary" blocks that make up the document. Often such blocks do not correspond to whole words, lines or paragraphs in the documents but just to fragments of words, thus a first aggregation based on their overlapping or adjacency is needed in order to obtain blocks surrounding whole words (*word-blocks*). Iteratively, 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*). To do this, our system exploits 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.

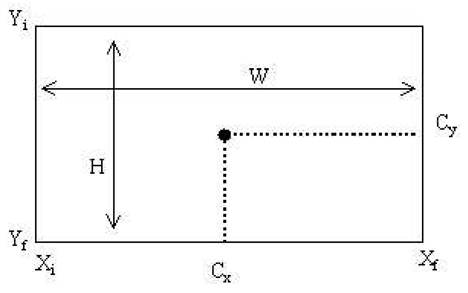


Figure 1. Block Features

Indeed, since grouping techniques based on the mean distance between blocks proved unable to correctly handle the case of multi-column documents, we turned to the application of 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. Specifically, such a learning task was cast to a Multiple Instance Problem and solved exploiting the kernel-based algorithm proposed in [12]. In our setting, each elementary block is described by means of a feature-vector of the form:

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

made up of parameters interpreted according to the representation in Figure 1, i.e.:

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

Starting with this basic 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 information about the name of the block New_Block_Name is a new name for the block in which is made in consideration the original names of both the original blocks, the information about the x and y positions are the original ones. The 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 2 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 adjacent block has to be considered an instance of positive examples, since they could be adjacent blocks but they could belong to a different frame than that of the considered block according to the original document. Such a situation is reported in Figure 3. 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 3) or when they belong to two different frames of the original document (for example, the *Title* and the *Authors* frame - left part of Figure 3).

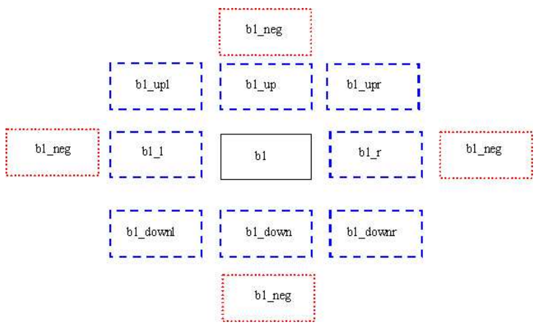


Figure 2. Close Neighbor blocks for block $b1$

In such a representation, a block O_n has at least one close neighbor block and at most eight (CNO_{nk} with $k \in \{1, 2, \ldots, 8\}$) (from top to bottom, from left to right - *top_left_corner, top, top_right_corner, right, down_right_corner, down, down_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 (MIP) to be solved by applying the Iterated-Discrim algorithm [12] in order to discover the relevant features and their values in this way obtaining 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, that represents the actual input to the next step in the layout analysis of the document is obtained.

In the following, an example of the representation is provided. Given the representation shown in Figure 2 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 "the block b35 can be merged with the blocks b36,b34, b24, b43 if and only if such blocks have the reported numeric features (dimensions 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]).
```

2.2. Discovery of the background structure of the document

The objects that made 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 para-

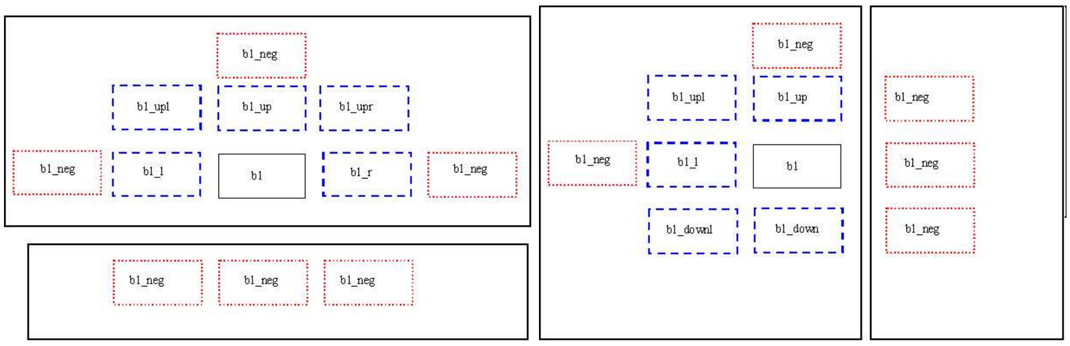


Figure 3. Selection of positive and negative blocks according the original document: one-column document on the left, two column document on the right

graph 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 electronic 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 [9] for addressing the key problem in geometric layout analysis. DOC analyzes the whitespace and background structure of documents in terms of rectangular covers, and it is efficient and easy to implement.

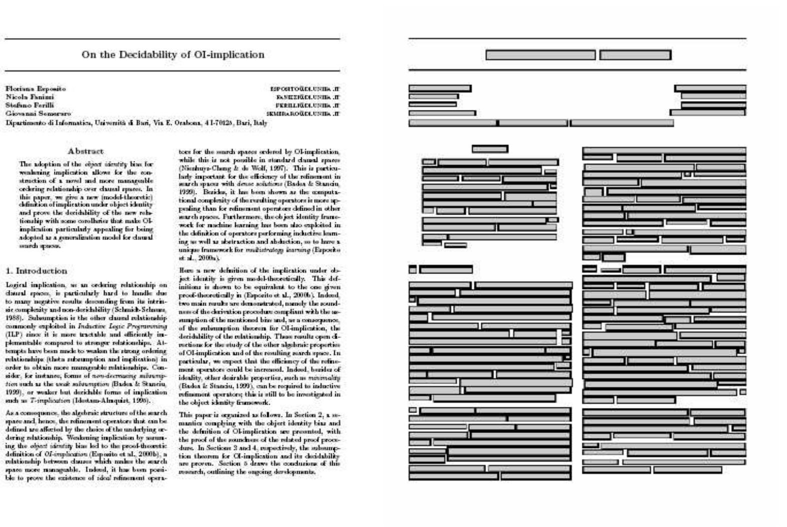
Once DOC has identified the whitespace structure of the document, thus yielding the background, it is possible to compute its complement, thus obtaining the document content blocks. When computing the complement, two levels of description are generated. The former refers to single blocks filled with the same kind of content, the latter consists in rectangular frames that may be made up of many blocks of the former type. Thus, the overall description of the document includes both kinds of objects, plus information on which frames include which blocks and on the actual spatial relations between frames and between blocks in the same frame (e.g., above, touches, etc.). This allows to maintain both levels of abstraction independently. The output of DOC is the *XML layout structure* (Figure 4 reports the graphic representation of the generated XML) of the original document, obtained through a process that is not a merely syntactic transformation from PS/PDF to XML.

It is wort to note that the exploitation of the pure algorithm reported in [9] in real document domains results unfeasible due to the great number of basic blocks discovered by the WINE tool. Indeed, by analyzing the document source, using WINE, it is possible to identify the basic blocks that make up the document. Often such blocks correspond to fragments of words, so that a first preliminary aggregation based on their *overlapping* or *adjacency* is needed in order to obtain blocks surrounding whole words. A further aggregation could be performed to have blocks that group words in lines by means of the procedure reported in Section 2.1. This procedure allows the DOC algorithm to be more efficient and effective. Moreover, 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 white space surrounding it 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 interword 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 $W(D)$ (computed by subtracting the sum of all the areas of the basic blocks from the whole area of the document $A(D)$) less than a given threshold δ, i.e.:
Let $A(R_i),\ i = 1, \ldots, n$ be the areas of basic blocks identified thus far in the document and $W(D) = A(D) - \sum_{i=1,\ldots,n} A(R_i)$, then the stop criterion s is established as:

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

The empirical study was performed applying the algorithm in full on a set of 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 5) normalized between 0 and 1, percentage of the last white area retrieved with re-



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 follow. 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.

Figure 4. Line and final layout analysis representations of the generated XML structure of a document

spect to the total white area of the current page of the document (bold line in Figure 5), percentage 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 5). The percentage 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 the 1/4 of the total steps of the algorithm. Such a consideration is generally valid for all the documents except for those that have a scattered appearance. Such a point, highlighted in the figure with a black rhomboidal shape, is the best stop point for the algorithm since before the layout is not very detailed while after it finds white spaces that are useless as shown with the black line in the graphic. Indeed, this is the point in which all the useful white spaces in the document, such as those between words, have been identified. Such a consideration is confirmed by analyzing the trend of the percentage of the last white area retrieved with respect to the total white area of the current page of the document (bold line) that decreases up to 0 in such a point. This suggests to stop the execution of the algorithm in such a point. It is worth noting that this value is reached very early, and before the size of the structure containing the blocks to be processed starts growing dramatically, thus saving lots of time and space resources.

3. Metadata Extraction

The availability of large, heterogeneous repositories of electronic documents is increasing rapidly, and the need for flexible, sophisticated document manipulation tools is growing correspondingly. These tools can usefully exploit

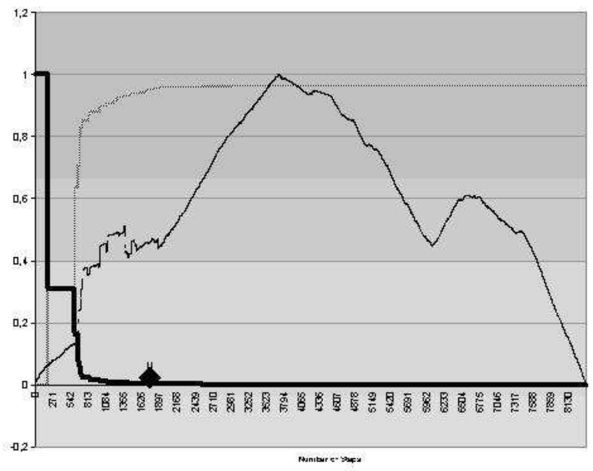


Figure 5. Stop Criterion Analysis

the logical structure, a hierarchy of visually observable organizational components of a document, such as paragraphs, lists, sections, etc. Knowledge of this structure can enable a multiplicity of applications, including hierarchical browsing, structural hyperlinking, logical component-based retrieval and style/format translation.

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 of the document to file it in the correspondent record, i.e. if the document is a magazine, or a book, or a scientific paper. Then, the significant components of the document have to be identified in order to extract the information needed to categorize it. Due to the huge amount of documents, carrying out such process manually is unfeasible. Here we suggest the use of a concept learning system to infer rules able to correctly classify the document type along with its significative components. The inborn complexity of document domain, and the need to represent relations among components, suggests the exploitation of symbolic first-order logic as a powerful representation language to handle such a situation. Specifically, 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 considered 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 documents of those new classes become available. This represents a problem for most existing systems, that require the set of classes for which definitions are to be learnt to be completely defined since the beginning.

Such considerations, among others on the learning systems available in the literature, led to the exploitation of INTHELEX (INcremental THEory Learner from EXamples) [14], whose most characterizing features are in its incremental nature, in the reduced need of a deep background knowledge, in the exploitation of negative information and in the peculiar bias on the generalization model, which reduces the search space and does not limit the expressive power of the adopted representation language.

3.1. The Learning System

INTHELEX is an Inductive Logic Programming [21] system that learns hierarchical logic theories from positive and negative examples: it is fully incremental (in addition to the possibility of refining a previously generated version of the theory, learning can also start from an empty theory), and adopts a representation language (Datalog^{OI} [26]) based on the Object Identity assumption, that 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, possibly related to each other; it can retain all the processed examples, so to guarantee validity of the learned theories on all of them.

3.1.1. The Inductive Core. INTHELEX is a closed loop learning system (i.e., a system in which feedback on performance is used to activate the theory revision phase [5]). The learning cycle it performs 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, namely training, tuning, and test examples, 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, then they are exploited to obtain a theory that is able to explain them. Such an initial theory can also be provided by the expert, or even be empty. Subsequently, the validity of the theory against new available examples, also stored in the example base, is checked by taking the set of inductive hypotheses and a tuning/test example as input and producing a decision that is compared to the correct one. In the 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. 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. In this way, tuning examples are exploited incrementally to modify incorrect theories according to a data-driven strategy. 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.

Whenever a new example is taken into account, it is stored in the historical memory of all past examined examples and the current theory is checked against it. If it 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 it covers the new example and is consistent with all the past negative examples. In such a case 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 if 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.

3.1.2. Multistrategy Features. Another peculiarity in INTHELEX is the integration of multistrategy operators that may help in the solution of the theory revision problem by pre-processing the incoming information, according to the theoretical framework for integrating different learning strategies known as Inferential Theory of Learning [20]. 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 some 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 [28], 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.

3.2. Representation Language

In order to exploit the learning system, a first-order logic representation of the document suitable for it must be provided. Thus, once the layout components of a document are automatically discovered as explained in the previous section, the next step concerns the automatic description of the pages, blocks and frames according to their size, spatial [24] and membership relations. Dealing with multi-page documents, we need to enrich the document description with *page information* such as: page number, whether the page is at the beginning, in the middle or at the end of the document, whether the page is the last one, total number of pages in the document. As pointed out, the automatic process ends with a set of rectangles in which each single page is divided. Such rectangles are described by means of their dimensions, 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 page and frame, e. g. $contain(page_i, frame_j)$, and between block and frame, e.g. $contain(frame_j, block_k)$.

Another possible relation between rectangles is the spatial one. Fixed a rectangle r, one can ideally divide the plan containing it in 25 parts (see Figure 6). Then, the relation between r and the other rectangles is described in terms of the occupied plans with respect to r. It is applied to all the blocks belonging to the same frame and to all the adjacent frames (The set of rectangles A that are adjacent to a rectangle r is made up of all the $r_i \in A$ such that r_i is the nearest rectangle of r in the same plan.). Moreover, such kind of representation of the plans allows the introduction in the example description of the topological relations [13, 24], including 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:

```
over_alignment(B1,B2):-
    occupy_plane_9(B1,B2),
    not(occupy_plane_4(B1,B2)).
over_alignment(B1,B2):-
    occupy_plane_10(B1,B2),
    not(occupy_plane_5(B1,B2)).
under_alignment(B1, B2) :-
   occupy_plane_19(B1, B2),
   not(occupy_plane_24(B1, B2)).
under_alignment(B1, B2) :-
   occupy_plane_20(B1, B2),
   not(occupy_plane_25(B1, B2)).
left_alignment(B1,B2):-
```

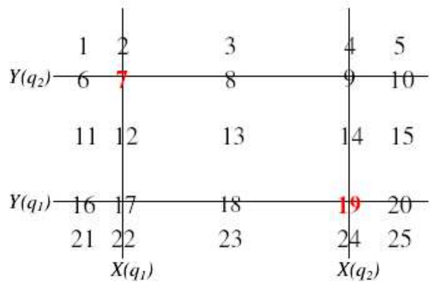


Figure 6. Representation Plans according to [24]

```
    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)).
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 7, form the initial representation that is made up, among other descriptors, by:

```
......
occupy_plane_18(b2, b1),
occupy_plane_19(b2, b1),
occupy_plane_20(b2, b1),
occupy_plane_23(b2, b1),
occupy_plane_24(b2, b1),
occupy_plane_25(b2, b1),
......
```

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

```
..., touch(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.

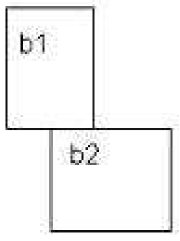


Figure 7. Block representation

3.3. Layout Correction

Due to the fixed stop threshold (see section 2.2), it could be the case that after the layout analysis step some blocks are not correctly recognized, i.e. background 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 and backward steps in the algorithm with respect to the stop threshold. This process 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 take 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, we decided to apply machine learning 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 we showed 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 the next layout analysis process. To this aim, a first-order description language and learning system were required, due to the need of expressing many relationships among blocks. Specifically, each example described the block to be forced

Page Descriptors	
page_number(d,p)	p is the number of page in the document d
last_page(p)	true if the page p is the last page of the document
in_fi rst_ pages(p)	true if the page p belongs to the fi rst n pages of the document (n < 1/3 total number of the pages in the document)
in_middle_pages(p)	true if the 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 the 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 that made up the document d
page_width(p,w)	w is the width of the page (a value normalized in [0,1])
page_height(p,h)	h is the height of the page (a value normalized in [0,1])
Frame/Block Descriptors	
frame(p,f)	f is a frame of the page p
block(p,b)	b is a block of the 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 width of the block in pixel
height(b,h)	h is the height of the block in pixel
x_coordinate_rectangle(r,x)	x is the coordinate of the start point of the rectangle (frame or block) r in horizontal sense
y_coordinate_rectangle(r,y)	y is the coordinate of the start point of the rectangle (frame or block) r in vertical sense
Topological Relation Descriptors	
belong(b, f)	the block b belongs to the frame f
pos_upper(p, r)	the rectangle r is positioned in the upper part of the page p (in vertical sense)
pos_middle(p, r)	the rectangle r is positioned in the middle part of the page p (in vertical sense)
pos_lower(p, r)	the rectangle r is positioned in the lower part of the page p (in vertical sense)
pos_left(p, r)	the rectangle r is positioned in the left part of the page p (in horizontal sense)
pos_center(p, r)	the rectangle r is positioned in the center part of the page p (in horizontal sense)
pos_right(p, r)	the rectangle r is positioned in the right part of the page p (in horizontal sense)
touch(b1,b2)	the block b1 touches the block b2 and vice versa
on_top(b1,b2)	the block b1 is positioned on the block b2
to_right(b1,b2)	the block b1 is positioned on the right of the block b2
over_alignment(b1, b2)	the block b1 is over the block b2
down_alignment(b1, b2)	the block b1 is under the block b2
left_alignment(b1, b2)	the block b1 is on the left of the block b2
right_alignment(b1, b2)	the block b1 is on the right of the block b2

Table 1. Attributes/Relations used to describe the documents

plus all the blocks around it, along with their size and position in the document, both before and after the manual correction.

3.4. Classification

After having detected the document layout structure, it is possible to associate a logic role with some of its components. In particular, this 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 different kinds of documents 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) paper 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. Here, the assumption is that documents belonging to the same class have a set of relevant and invariant layout characteristics (*page layout signature*). Again, the different and complex relationships that hold between the layout component 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, in addition to new instances of available classes, asked for incremental abilities that INTHELEX provides.

Classification of multi-page documents is performed by matching the layout structure of the first page against models of classes of documents. These models capture the invariant properties of the images/layout structures of documents belonging to the same class. They are 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 answers to the requirement of flexibility and generality.

3.5. Understanding

Once the class of a document has been identified on the basis of its first page, its logical components that are present in all pages can be located and tagged by matching the layout structure of the 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, 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 depend 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.

4. Categorization, Filing and Indexing

One of the most important task in a digital library management concerns the categorization of the documents. The effectiveness in performing this task represents the success factor in retrieval process of the documents that are really interesting for the users. Indeed, a problem of most existing word-based retrieval systems consists of their ineffectiveness in finding interesting documents when the users do not use the same words 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 may use a different word than the one that appears in a document, and may not retrieve the document; The same word can have multiple meanings (*polysemy*), so a searcher can get unwanted documents with the alternate meanings.

The solution to such problems is represented by the Latent Semantic Indexing (LSI) technique [10] 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. We exploit the LSI in both the indexing and retrieval processes as in the following explained.

In the indexing process, LSI uses a term-document matrix which describes the occurrences of terms in documents; it is a sparse matrix, since not all the terms are present in all documents, whose rows correspond to documents and whose columns correspond to terms, typically stemmed words that appear in the documents, specifically, the elements of the matrix represent the frequency of the times the terms appear in each document. Obviously, the absence of a term in a document is represented by the 0 value in the matrix. The next step concerns the weighting of the matrix that allows to create a correspondence between the local (frequency of the term i in the document j (TF), tf_{ij}) and the global (frequency of the term i in the whole set of documents (IDF), gf_i) relevance for each term.

A good weighting function is obtained by the combination of both the local and global relevance of the term:

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

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 relation such values is represented by the log entropy function, where:

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

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

N represents the number of the documents and $p_{ij} = \frac{tf_{ij}}{gf_i}$. This way, the logarithmic value of the local factor $L(i,j)$ mitigates the effects due to great variations in term frequencies, while the entropy function of the global factor $G(i)$ mitigates the noise that could be present in the documents.

After the construction of the weighted occurrence matrix, LSI finds a low-rank approximation to the term-document matrix. The reasons for the approximations can have two explanations: The original term-document matrix is supposed to be too large for the computing resources; The original term-document matrix is supposed to be *noisy*: for instance, anecdotal instances of terms are to be eliminated. From this point of view, the matrix is interpreted as a *de-noisified matrix* (a better matrix than the original). Concretely, the downsizing of the matrix is achieved through the use of singular value decomposition (SVD): the set of all the terms is then represented by a vector space of lower dimensionality than the total number of terms in the vocabulary. The consequence of the rank lowering is that some dimensions get 'merged': {(car), (truck), (flower)} $\rightarrow$ {(1,3452 $*$ car + 0,2828 $*$ truck), (flower)} This mitigates polysemy, as the rank lowering is expected to merge the dimensions associated to terms of similar meanings.

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 concerns the retrieval phase, in similar way, firstly, it is built a vector for the query which describes the occurrences of terms, previously identified for the documents, in the query. Once the vector is created, a weighting function is applied to each element e of the 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 whole set of documents and n is the number of documents in which the term i appears.

As regards the reduction of the query to a k-dimension vector, it is performed by exploiting the SVD decomposition of the term-document matrix previously computed. This process is carried out for each vector representing the document in the weighting term-document matrix too. At the end of these steps both the query and the documents are k-dimension vectors. It is worth noting that this step is necessary in order to make comparable the query and the documents vectors that will be performed by means of a similarity function. In our system the *cosine similarity* function [4] was exploited to perform the comparison between the query vector and each document vector. The documents that have a high degree of similarity according to the value computed are the documents 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 [7] and SVD-Updating [23]. The former is a much simpler alternative that uses the existing SVD to represent new information but yields poor-quality updated matrices, since the information contained in the new documents/terms is not exploited by the updated semantic space. The latter, that is exploited in our system, represents a trade-off between the former and the recomputation from scratch.

5. Exploitation and Evaluation

The system for automated electronic 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.

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.

In this scenario the first step concerns the document image classification and understanding of the submitted documents by Authors. In order to evaluate the system on this phase, experiments were carried out on a dataset made up of 108 documents (scientific papers) coming from online repositories and formatted according to the Springer-Verlag Lecture Notes in Computer Science (`LNCS`) style, the Elsevier journals style (`ELSEVIER`), the International Conference on Machine Learning (`ICML`) and European conference on artificial intelligence (`ECAI`) proceedings style. Specifically, 31 papers were formatted according to `LNCS` style, 32 according to `ELSEVIER` style, 20 according to `ICML` style and 25 according to `ECAI` style.

We chose papers formatted according to different styles to show the ability of the incremental system in learning 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 we may build a global theory, containing the definitions of different document styles, that should be used for many conferences.

Each document was pre-processed and automatically described in a first-order logic language, along the features reported in previous section. Each document instance obtained in such a way was considered a positive example for the class it belongs to, and as a negative example for all the other classes to be learned. The system performance was evaluated according to a 10-fold cross validation methodology, ensuring that all the learning and test sets contained the same proportion of positive and negative examples. Furthermore, the system was provided with background knowledge expressing topological relations (see Section 3.2), and abstraction was exploited to discretize numeric values of size and position. In the following an extract of the abstraction rules for rectangles width discretization are 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.
```

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

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

Label	Revisions	RunTime (sec.)	Accuracy %
title	11.29	33.467	95.93
author	12.27	47.88	95.39
abstract	16.48	133.706	93.06
references	13.29	50.94	95.24

Table 3. Learning System Performance: inferring rules for the identification of logical components of `LNCS`

In general, due to the different layout components that could be interesting for a specific class of documents but not for others, the image understanding phase must be preceded by a classification process in order to recognize the correct class the document belongs to. Hence, a first experiment on the inference of rules to be exploited during the classification step was run, showing good system performance in terms of execution time, predictive accuracy and number of theory revisions, as reported in Table 2. The lowest accuracy refers to `LNCS`, which could be expected due to the less strict check for conformance to the layout standard by the editor.

The second experiment, concerning the image understanding phase, aimed at learning rules able to identify the layout components, was performed on the *title*, *authors*, *abstract* and *references* layout components of the documents belonging to the `LNCS` class. This class was chosen since it represent the layout of the 264 papers submitted to the 18th Conference on Industrial & Engineering Applications of Artificial Intelligence & Expert Systems (IEA/AIE 2005) used for further experiments in the following reported. In Table 3 the averaged results of the 10 folds along with the execution time, predictive accuracy and number of theory revisions are reported. As we can note, the system showed good performance, specifically as regards the references on which we intend to base additional processing steps.

One of the hardest and most time-consuming tasks in Scientific Conferences organization is the process of assigning reviewers to submitted papers. Due to the many constraints to be fulfilled, carrying out manually such a task is very tedious and difficult, and does not guarantee to result in the best solution. This complex real-world domain can take advantage of the proposed document management system as concerns both the indexing and retrieval of the documents and their associated topics. In the following we present an experiment carried out on a dataset composed by the 264 papers submitted to the 18th Conference on Industrial & Engineering Applications of Artificial Intelligence & Expert Systems (IEA/AIE 2005), whose Call for Papers identified 34 topics of interest.

Firstly, the layout of each paper in electronic 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. On such a text the LSI technique was exploited as in the following reported. The words contained therein were stemmed according to the technique proposed by Porter [25], resulting in a total of 2832 word stems, on which the LSI technique was applied in order to index the whole set of documents. Then, the same procedure was applied to index the reviewers, resulting in 2204 stems. In this case, the titles of their papers appearing in the DBLP Computer Science Bibliography repository (`http://www.informatik.uni-trier.de/~ley/db/`) were exploited. With respect to exploiting their homepages' information on research interests, this ensured a more uniform description. Compared to manually selecting the title of their publications, this ensured more completeness, even if at the cost of not having the abstracts available as well.

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 in performing 34 queries, each corresponding to one conference topic, on both the papers and the reviewers in the database previously indexed, and then in associating to each paper/reviewer the topics for which it/he appears among the first l results. Specifically, the results on document topic recognition showed that 88 documents per query had to be considered, in order to include the whole set of documents. However, returning just 30 documents per query, 257 out of 264 documents (97.3%) were already assigned to at least one topic, which is an acceptable trade-off (the remaining 7 documents can be easily assigned by hand). Thus, 30 documents were considered a good parameter, and exploited to count the distribution of the topics between the documents. Interestingly, more than half of the documents (54.7%) concern between 2 and 4 topics, which could be expected both for the current interest of the researchers in mixing together different research areas and for the nature of the topics, that are not completely disjoint (some are specializations of others). Evaluated by the conference organizers, the result showed a 79% accuracy on average. As to the reviewers, even if taking $l = 6$ already ensured at least one topic for each of them, we adopted a more cautious approach and took $l = 10$, in order to balance the possible inaccuracy due to considering only the titles of their publications. The resulting accuracy was 65%.

Lastly, the topics automatically associated to papers and reviewers were fed to the expert system `GRAPE` [19] in order to perform the associations, with the requirement to assign each paper to 2 reviewers. `GRAPE` (Global Review Assignment Processing Engine), is an expert system, for solving the reviewers assignment problem, that takes advantage of both the papers content (topics) and the reviewers expertise and preferences (biddings). It could be used by exploiting, in addition to the papers topics, the reviewers expertise only, or both the reviewers expertise and biddings.

In order to have an insight on the quality of the results, in the following we present some interesting figures concerning `GRAPE`'s outcome. In solving the problem, the system was able to complete its task in 2 minutes. `GRAPE` was always able to assign papers to reviewers by considering the topics only, except in two cases. In particular, except for those reviewers that explicitly asked to review less than 10 papers (*MaxReviewsPerReviewer* constraint), it assigned 10 papers to 40 reviewers, 9 to 2 reviewers, 8 to 3 reviewers, 7 and 6 to one reviewer. The experts considered the final associations made by GRAPE very helpful, since they would have changed just 7% of them.

6. Conclusion

The huge amount of documents available in electronic form and the flourishing of digital repositories raises problems concerning document management, aimed at effectiveness and efficiency of their successive retrieval, that cannot be faced by manual techniques. This paper proposed the intensive application of intelligent techniques as a support to all phases of automated document processing, from document acquisition to document indexing and retrieval. Experiments in the real-world sample application domain of automatic Scientific Conference Management prove the viability of the proposed approach.

Different future work directions are planned for the proposed system. First of all, the automatic processing of the bibliographic references that can improve the identification of the subject and context of the document. In this respect, some initial research is already present in the literature [8], and we are currently working on the improvement of such methods, and on the development of new, knowledge-based ones. More future work will concern the use of ontologies for improving matching effectiveness.

References

[1] Adobe Systems Inc. *PostScript language reference manual – 2nd ed.* Addison Wesley, 1990.

[2] Adobe Systems Inc. *PDF Reference version 1.3 – 2nd ed.* Addison Wesley, 2000.

[3] M. Aiello, C. Monz, and L. Todoran. Document understanding for a broad class of documents. *International Journal on Document Analysis and Recognition (IJDAR)*, 5(1):1–16, 2002.

[4] R. A. Baeza-Yates and B. A. Ribeiro-Neto. *Modern Information Retrieval*. ACM Press / Addison-Wesley, 1999.

[5] J. Becker. 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.

[6] T. Berners Lee, J. Hendler, and O. Lassila. The semantic web. *Scientifi c American*, 284(5):34–43, 2001.

[7] M. W. Berry, S. T. Dumais, and G. W. O'Brien. Using linear algebra for intelligent information retrieval. *SIAM Rev.*, 37(4):573–595, 1995.

[8] D. Besagni and A. Belaïd. Citation recognition for scientifi c publications in digital libraries. In *Proceedings of the First International Workshop on Document Image Analysis for Libraries (DIAL 2004)*, pages 244–252, 2004.

[9] T. M. Breuel. Two geometric algorithms for layout analysis. In *Workshop on Document Analysis Systems*, 2002.

[10] S. C. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman. Indexing by latent semantic analysis. *Journal of the American Society of Information Science*, 41(6):391–407, 1990.

[11] A. Dengel. Making documents work: Challenges for document understanding. In *Proc. of Seventh International Conference on Document Analysis and Recognition (ICDAR)*, pages 1026–1037, 2003.

[12] T. G. Dietterich, R. H. Lathrop, and T. Lozano-Perez. Solving the multiple instance problem with axis-parallel rectangles. *Artifi cial Intelligence*, 89(1-2):31–71, 1997.

[13] M. Egenhofer. Reasoning about binary topological relations. In O. Gunther and H.-J. Schek, editors, *Second Symposium on Large Spatial Databases*, volume 525 of *Lecture Notes in Computer Science*, pages 143–160. Springer, 1991.

[14] F. Esposito, S. Ferilli, N. Fanizzi, T. M. Basile, and N. Di Mauro. Incremental multistrategy learning for document processing. *Applied Artifi cial Intelligence: An Internationa Journal*, 17(8/9):859–883, September-October 2003.

[15] F. Esposito, D. Malerba, G. Semeraro, S. Ferilli, O. Altamura, T. M. A. Basile, M. Berardi, M. Ceci, and N. D. Mauro. 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)*, pages 328–335, 2004.

[16] W. Glunz. pstoedit - a tool converting postscript and pdf fi les into various vector graphic formats. (http://www.pstoedit.net).

[17] A. Kakas and P. Mancarella. On the relation of truth maintenance and abduction. In *Proceedings of the 1st Pacifi c Rim International Conference on Artifi cial Intelligence*, Nagoya, Japan, 1990.

[18] J. Kim, D. Le, and G. Thoma. Automated labeling in document images. In *Document Recognition and Retrieval VIII - Proceedings of The International Society for Optical Engineering*, volume 4307, pages 111–122, 2001.

[19] N. D. Mauro, T. M. A. Basile, and S. Ferilli. Grape: An expert review assignment component for scientifi c conference management systems. In *Innovations in Applied Artifi cial Intelligence: 18th International Conference on Industrial and Engineering Applications of Artifi cial Intelligence and Expert Systems (IEA/AIE 2005)*, volume 3533 of *Lecture Notes in Computer Science*, pages 789–798. Springer Verlag, 2005.

[20] R. Michalski. Inferential theory of learning. developing foundations for multistrategy learning. In R. Michalski and G. Tecuci, editors, *Machine Learning. A Multistrategy Approach*, volume IV, pages 3–61. Morgan Kaufmann, 1994.

[21] S. Muggleton and L. De Raedt. Inductive logic programming: Theory and methods. *Journal of Logic Programming*, 19/20:629–679, 1994.

[22] G. Nagy. Twenty years of document image analysis in PAMI. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(1):38–62, 2000.

[23] G. W. O'Brien. Information management tools for updating an SVD-encoded indexing scheme. Technical Report UT-CS-94-258, University of Tennessee, 1994.

[24] D. Papadias and Y. Theodoridis. Spatial relations, minimum bounding rectangles, and spatial data structures. *International Journal of Geographical Information Science*, 11(2):111–138, 1997.

[25] M. F. Porter. An algorithm for suffi x stripping. In J. S. Karen and P. Willet, editors, *Readings in information retrieval*, pages 313–316. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1997.

[26] G. Semeraro, F. Esposito, D. Malerba, N. Fanizzi, and S. Ferilli. A logic framework for the incremental inductive synthesis of datalog theories. In N. Fuchs, editor, *Proceedings of the 7th International Workshop on Logic Program Synthesis and Transformation*, volume 1463 of *LNCS*, pages 300–321. Springer, 1998.

[27] S. L. Weibel, M. Oskins, and D. V. Goetz. Automated title page cataloging, a feasibility study. *Information Processing and Management*, 25(2):187–203, 1989.

[28] J.-D. Zucker. Semantic abstraction for concept representation and learning. In R. S. Michalski and L. Saitta, editors, *Proceedings of the 4th International Workshop on Multistrategy Learning*, pages 157–164, 1998.