Information Capture and Semantic Indexing of Digital Libraries through Machine Learning Techniques

Floriana Esposito *Donato Malerba* *Giovanni Semeraro*
Cesare Daniele Antifora *Gioacchino de Gennaro*

Dipartimento di Informatica - Università degli Studi - via Orabona, 4 - 70126 Bari, Italy
{esposito | malerba | semeraro}@lacam.uniba.it *{cesare | nino}@minsky.uniba.it*

Abstract

This paper presents a prototypical digital library service. It integrates machine learning tools and techniques in order to make effective, efficient and economically feasible the process of capturing the information that should be stored and indexed by content in the digital library. In fact, information capture is one of the main bottleneck when building a digital library, since it involves complex pattern recognition problems, such as document analysis, classification and understanding. Experimental results show that learning systems can solve effectively and efficiently all these problems.

1: Introduction

Building digital libraries is one of the most important Grand Challenge problems faced by information professionals [6]. A digital library is *"a distributed technology environment which dramatically reduces barriers to the creation, dissemination, manipulation, storage, integration and reuse of information by individuals and groups."* [8].

The main functions of a digital library, as well as of a conventional library, can be summarized as: 1) Collection; 2) Organization and representation; 3) Access and retrieval [7]. *Collection* does not mean acquisition of information items from any information resource, but includes techniques that allow to detect those information sources that are useful to a client population. *Organization* and *representation* require that the information resources are classified and indexed according to criteria relevant to their potential users. *Access* and *retrieval* involve the design and organization of materials within a physical space in order to retrieve effectively information items, when the user requires them.

Building an effective and efficient digital library service is the task of a project that we started recently, as the natural evolution of an erlier project on automated document processing [3]. Since the beginning, it has been clear that the key issue of the project was system integration. This means to develop an architecture that allows an effective integration of tools for information compression, storage, organization, retrieval and navigation, as well as with friendly and world-wide available standard graphical user interfaces (GUIs), and multimedia technology. One of the peculiarities of our project lies in the role that intelligent agents, namely learning systems, can play for *information capture* and *semantic indexing*.

By *information capture*, we mean the task of setting information items free of the physical medium on which they are stored. It involves the problem of converting data from a paper format into a digital one. This has given raise to the birth of new research areas dealing with the automated recognition of the components of documents, ranging from low-level elements, such as characters (OCR), symbols, text, lines, to high-level ones, such as sections, columns, graphics, images, handwriting [15]. Several problems are addressed by these new areas, namely the analysis of the overall physical structure of a document (*document analysis*), the classification of the whole document (*document classification*), and the analysis of the overall logical structure of a document (*document understanding*). All these problems aim at the high-level understanding of documents. In fact, this issue is a *conditio sine qua non* to index the information in the library according to its content (*semantic indexing*).

This paper presents a prototypical intelligent system, that integrates learning agents for document analysis, classification and understanding into a digital library service. The structure of the paper is as follows. In the next section, the role of machine learning within the digital library service is investigated and depicted. Section 3 presents the architecture of IDL (Intelligent Digital Library), a prototype of an intelligent digital library service. Section 4 reports the plan of the future work on IDL.

2: Machine learning for information capture and semantic indexing

A major bottleneck in developing technologies for digital libraries is the transformation of data presented on paper into an effective electronic representation. Indeed, the non-existence of economically feasible capabilities to replace existing books with their digital counterparts was the most important reason for the partial failure of the National Library of Medicine project [12]. A solution to this problem could come from the application of document processing techniques [13]. Since the pixel representation of a document image is far from being considered *effective*, it is necessary to process the bitmap in order to extract an abstract representation of the content of the document itself.

Information capture through document processing can take place at several levels of abstractions. At the lowest level, called *document analysis*, it aims at extracting the *geometric* (or *layout*) structure of a document, that is, a hierarchical organization of layout areas with contents of different categories (text, image or graphics). The segmentation of the document image into *primitive tokens* (rectangular *blocks* associated with only one category) and the grouping of tokens in order to form larger and meaningful layout components, are some examples of tasks performed by document analysis systems.

After detecting the layout structure, the logical constituents of the document, such as title, authors, sections of a paper, may be identified. This high level process is called *document understanding*, since logical constituents are much closer to semantics than the layout components. The logical objects can be arranged in another hierarchical structure, which is called *logical structure*. Generally, it is possible to find some correspondences between the two structures of a document, in which case the document understanding process can be viewed as a mapping from the layout into the logical structure. This activity mimes a typical behaviour of humans, who are often able to associate a possible meaning to some tokens without reading inside, but using only information on the spatial organization of blocks in the page layout. Nevertheless, when the document processing system has to deal with a variety of layout and logical structures, it becomes difficult to find all correct mappings. In this case, the problem can be simplified by *classifying* the documents into a set of distinct classes, each of which is characterized by certain standard layout and logical structures (*document classification*).

To sum up, in document processing, we can distinguish three main tasks: Document analysis, document classification and document understanding [3]. We argue the possibility that the accomplishment of all such tasks can actually benefit by the application of machine learning techniques. Thus, the goal of automatically capturing information from documents takes advantage of some AI techniques, which have often been pointed out as a solution to the knowledge acquisition bottleneck.

A first step towards the organization of the page layout consists of the separation of text from graphics, that is, the association of a semantic primitive concerning the type of content to the set of tokens detected by the segmentation process. In particular, we defined five basic categories of blocks, namely text, picture, graphic, horizontal and vertical solid black line, which allow us to distinguish larger layout components such as paragraphs, tables, line drawings, halftone images and formulas. Each block is described by ten different numerical features automatically produced by the segmentation module (height, length, area, eccentricity of the block,...). All these features concern global properties of the blocks and not the configurations of pixels within them. Therefore, the problem is that of classifying each block on the ground of these geometric/pictorial features, that is, without "watching" or "reading" its content. We induced a decision tree from a set of 52 training documents, whose blocks have been previously classified by hand. The best result we observed is an average accuracy above 97%.

Once the type has been associated to the primitive tokens, meaningful groupings of blocks are built on the ground of both perceptual/spatial criteria and knowledge about typesetting conventions [9]. The reason for detecting structures among tokens (*layout analysis*) is to reduce the computational complexity of the subsequent recognition process.

The computational strategy adopted for understanding a document consists of a hierarchical model fitting, which limits the range of labelling possibilities by working hierarchically. More precisely, the page layout of a document is described by means of a first-order logic language, and such description is first matched against models of classes of documents and then against models of the single logical components of interest for that class. All models are expressed in the same language used to describe the page. The choice of a first-order logic language answers to the requirement of flexibility and generality. Once again, models used in this computational strategy have been automatically induced from the first-order descriptions of the layout of some training documents. More precisely, each training document is associated with a class and a set of labelled layout components. Therefore, two learning problems can be defined: First, induce models of classes of documents and then find models of the logical structures of each class. The former is useful for the task of document classification while the latter is applied to the document understanding phase.

Several approaches have been investigated in order to solve these two learning problems. Initially, a supervised inductive learning system that implements a hybrid approach was applied to the document classification task [1]. In this experiment, the layout of each document was described by numeric and symbolic features; a parametric method for linear classification used the numeric features, while a conceptual method induced some models from the symbolic features alone. The combination of the predictions made by the two methods provided the best results in terms of simplicity and predictive accuracy of the learned models. This hybrid approach, however, operates in a batch way: All models are learned from scratch each time the learning process is activated. A more recent experimentation has shown promising results also for an incremental approach in which models are progressively specialized and generalized as new incoming documents are misclassified or not classified at all (theory revision) [11].

As to the document understanding task, a different approach has been tested. In this case, indeed, we have to learn models of the logical components which refer to a part of the document rather than to the whole document. Nevertheless, logical components of a document may be related to each other, as in the case of standard scientific papers, where the author's affiliation is above the abstract and under the title. Thus, it would be more appropriate to learn models that reflect these dependencies among components of the logical structure. This hypothesis has been tested by inducing both independent and dependent models of the logical components. In the latter case, the learning system was provided with a dependency graph of logical components and performed a shift of the languages of observations and hypotheses in order to generate dependent models [10]. From the experimental results we draw two main conclusions: First, taking into account the dependencies between logical components improves the accuracy and the comprehensibility of the models as well as the efficiency of the learning process; second, it is possible to rely on statistical techniques in order to discover such dependencies before starting the induction process.

3: The architecture of the digital library

The learning agents that perform the tasks of layout analysis, document classification and document understanding constitute three *application enablers* in the system architecture of IDL, a prototypical intelligent digital library service. Indeed, a *digital library service* is a set of modules that can be classified as either resource managers or application enablers. A *resource manager* is a program that represents the only access path to the data contained in a *protected resource* and is accessible to multiple, concurrent clients. Intuitively, a protected resource is a data collection. An *application enabler* is a software that allows a class of users to make application programming easy and quick (or to avoid it completely).

The architecture of IDL is shown in Fig. 1. The *Repository* is a protected resource containing the actual collection of data that constitutes the digital library. Usually, it consists of highly structured items. Here, we use the word *database* rather than *information collection* or *knowledge database* because in our prototype it is an object-oriented database, since we made use of a commercial object-oriented database management system, called ObjectStore by Object Design, Inc. Thus, the Repository is actually a set of objects. More precisely, these objects constitute an instance of an ObjectStore conceptual scheme, that we designed purposely for the digital library. The *document object model* is the conceptual schema according to which documents are stored and internally represented in both the layout and the logical structure. The

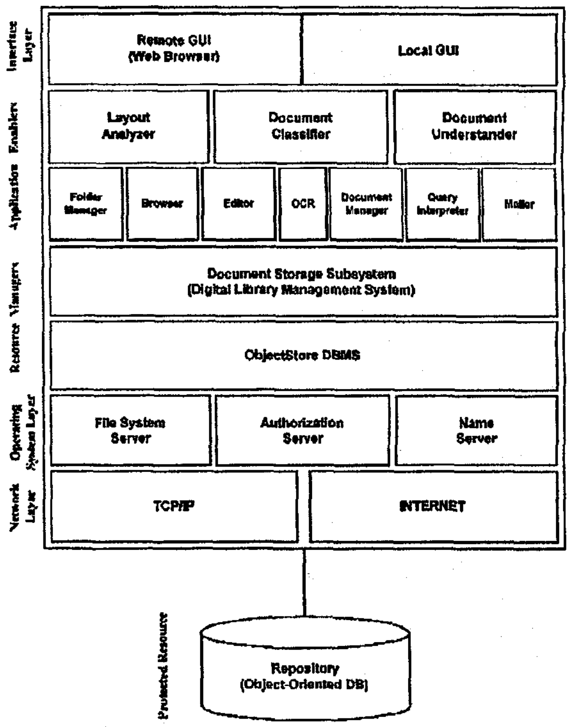


Figure 1. The architecture of the digital library service.

layout structure associates the content of a document with a *hierarchy of layout objects*, such as text lines, vertical/horizontal lines, graphic elements, images, columns and pages. The leaves of a *layout tree* representing a hierarchical geometrical structure are called *basic blocks*, and they typically correspond to rectangular areas which delimit portions of contents on the presentation medium. All the internal nodes of the layout tree are named *composite objects* since they are obtained by grouping together basic blocks as well as composite objects of lower level. The root of the layout tree represents the whole document. In multi-page documents, such as books, a composite object can represent a set of pages, where a *page* is a rectangular area that corresponds to a unit of the presentation medium. All other internal blocks are named *frames* and correspond to rectangular areas within a page. The portion of the object model that allows us to store (and retrieve) the objects related to the layout structure of a document is given in Fig. 2a. For our purposes, we defined five distinct levels of objects in the layout tree (other than `Page`), called `BasicBlock`, `LineBlock`, `SetOfLineBlock`, `FirstFrameBlock`, and `SecondFrameBlock`.

Each level corresponds to an internal representation of the layout structure of a document at a different level of granularity (levels are listed starting from the coarsest one - `BasicBlock` - to the finest one - `SecondFrameBlock`) and every object at a level is made up of objects at a lower level. All the objects at any level have the same basic structure, since they are instances of the class `Block` (in Fig. 2a), as well.

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. Generally, this structure associates the contents of the document with a *hierarchy* of *logical objects* such as title, authors, abstract, paragraphs, footnotes, and so on. Even for the logical structure, it is possible to distinguish *basic logical objects*, which appear at the bottom of the logical tree, from the *composite logical objects*, which are represented as internal nodes of the logical tree. Obviously, the types of logical objects in a document are strongly application dependent, thus for a business letter it is sensible to look for sender, receiver, date, logotype, reference number, body and signature, while for a scientific paper it is possible to define title, authors, affiliation, keywords, abstract, subtitle, paragraph, header, footnote, page number and caption [14]. This is not so for the layout objects, that is to say, the layout structure does not vary according to the type of document we are dealing with (scientific paper or business letter).

The information about the type of document is stored in the portion of the document object model depicted in Fig. 2b. Specifically, each class of documents in the digital library is defined as an instance of the object `DocClass`. In fact, defining each class of documents as an instance of a meta-level class - `DocClass` - allows us to achieve a greater flexibility when the digital library needs to be updated. Moreover, as previously stated, each class of documents is associated with a set of meaningful types of logical objects, called *logical labels* (title, authors, affiliation, keywords, abstract for scientific paper), thus adding a new class of documents requires the introduction of a set of new logical labels. This is performed by creating a new instance of the class `Attribute` for each logical label (Fig. 2b).

With reference to Fig. 1 again, above the protected resource, there is the digital library software. It consists of five layers. The lower layers are those more related to the machinery used to implement the digital library and ignore the semantics of the repository, since they do not need to know the format of the data stored in it. The *NetworkLayer* allows remote access to the library via Internet, while the *Operating System Layer* makes available all the functions of the operating system and controls that the users who required an access to the repository have proper access rights. In detail, it maps names of the users that issued the request to the proper locations by means of the *Name Server*, and limits each user to what the owner of the digital library (*library's custodian*) permits by means of the *Authorization Server*.

The layer of the *Resource Managers* mainly deals with the management of documents, viewed at different levels of abstraction. The lower box in this layer - *ObjectStore DBMS* - is a database management system. The upper box contains the *Document Storage Subsystem*, that is to say, the document storage and access software. It is involved in both storing and retrieving items to and from the library collection and updating and searching the library catalogs. It is worth to say that its scope is limited to aspects that are independent of the meaning and the internal representation of information items in the digital library. It is implemented as a client-server tool.

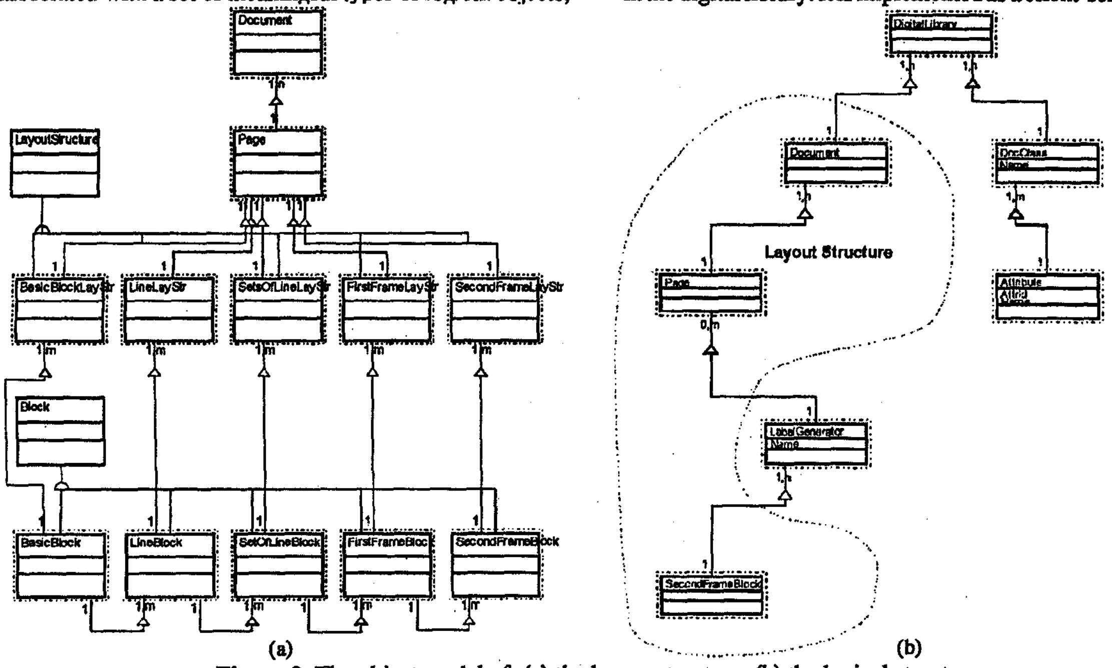


Figure 2. The object model of: (a) the layout structure; (b) the logical structure.

In [7], it is defined as *"a resource manager that creates a library abstraction implementing all the essential data storage, retrieval, protection, communication, and search functions as primitive operations which do not depend on the interpretation of the data handled."*

The layer of the *Application Enablers* makes available several functionalities to the end users of the library. This layer hides operating system and machine differences. The *Mailer* enabler implements a standard electronic mailing system. The *Document Manager* is in charge of helping end users with their special kinds of documents, mainly as regards their presentation and manipulation. The *Query Interpreter* is the inference engine that allows the user to formulate any query concerning the objects in the library. The query language interpreted by this enabler is a first-order logic language, whose primitives are:

i) (<OBJECT> **IS IN** <CLASS>)

ii) (**THE** <ATTRIBUTE> **OF** <OBJECT> **IS** <VALUE>)

where <OBJECT>, <CLASS>, <ATTRIBUTE> and <VALUE> are metasymbols that can be properly replaced by variables or constants in each query. The former kind of primitives, called *Class Query*, allows to answer to membership class queries, such as *"List all the documents in the class of scientific papers"* or *"Is the document #101 an instance of the class of business letters?"*, while the latter, called *Attribute Query*, is useful to find objects whose attributes meet specific conditions about their values. Examples of Attribute Queries are *"List all the papers whose author is John Smith"* or *"Who are the authors of the paper #101?"*. A query is a proper combination of these two kind of primitives through the logical operators **AND**, **OR**.

The *Browser* enabler is a tool that allows the user to navigate into the digital library. It is intended to be exploited mainly by people who do not know the organization of the library, just like someone using a library for the first time. The *Folder* enabler is used to create new folders, add a document to a folder, delete an existing folder. The *Editor* enabler is activated when a user wants to change a document. Typically, this is possible on local copies of a document, unless the user is the library's custodian. The upper layer of application enablers includes the tools for document layout analysis, document classification and document understanding. As told in the previous section, these tools are used to recognize patterns and layout/logical structures of the documents in the library, with the aim to create search indexes automatically [2;4]. The *OCR* enabler is a classical OCR system, integrated into IDL with the aim of "reading" the content of all and only those layout blocks of a document that are considered useful to the semantic indexing and retrieval by content of the document itself.

The *Interface Layer* implements the applications that actually interface the users of the library. Currently, both a local GUI running on Solaris 1.0 on a Sun SPARCstation 10, and a remote GUI have been developed. The former is intended to be used by the library's custodian, while the latter is the interface for the end users of the library, who can only browse or query the digital library on the ground of the content of its documents. Both of them are based on any Web browser and are designed around a state-transition model (finite state automata), with each state representing an HTML page. Fig. 3 shows the schema of generation for the HTML pages of the remote GUI. All the HTML pages are dynamically generated by Common Gateway Interface (CGI) scripts (in C language) in order to reflect the current content of the digital library. Some HTML pages, corresponding to types 1, 4 - 6 in Fig. 3, are reported in Fig. 4.

4: Conclusions and future work

Machine learning techniques can offer an added value when building intelligent digital libraries. Indeed, all the tasks related to the information captureand semantic indexing can take advantage of the use of learning agents for layout analysis, document classification and understanding. In the paper, we have presented a prototype of an intelligent digital library service, and we have shown where and how digital libraries can benefit by the use of machine learning techniques. Future work will concern the extension of the digital library tools and services in order to deal with different kinds of documents, such as topographic maps for applications like geographic information systems [5], and technical documents for applications that support project development.

References

[1] Esposito, F., D. Malerba, G. Semeraro, E. Annese, and G. Scafuro (1990). An experimental page layout recognition system for office document automatic classification: an integrated approach for inductive generalization. *Proc. 10th Int'l Conf. on Pattern Recognition*, 557-562. Los Alamitos:

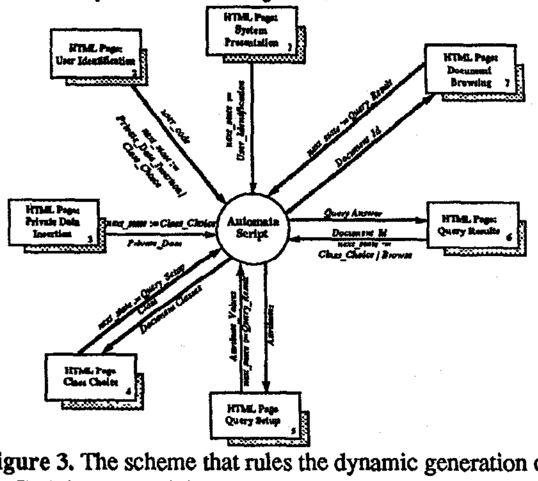
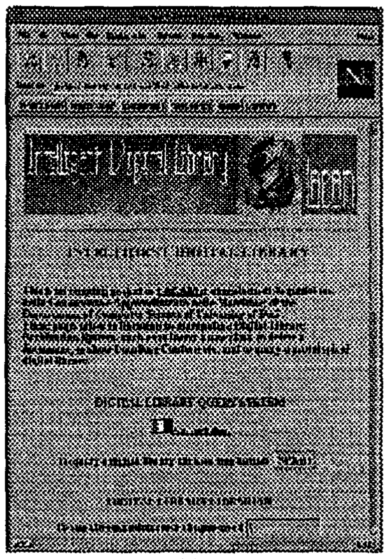
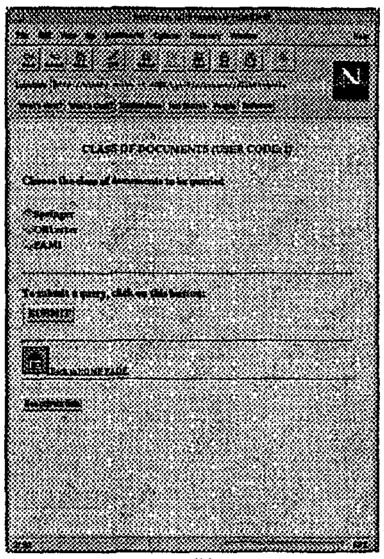

HTML Page: System Presentation 1
HTML Page: User Identification 2
HTML Page: Document Browsing 7
user_Code
next_state := Private_Data_Insertion | Class_Choice
next_state := User_Identification
next_state := Query_Results
Document_Id
HTML Page: Private Data Insertion 3
next_state := Class_Choice
Private_Data
Automata Script
Query Answer
Document_Id
next_state := Class_Choice | Browse
HTML Page: Query Results 6
next_state := Query_Setup
Class
Document Classes
Attribute Values
next_state := Query_Results
Attributes
HTML Page: Class Choice 4
HTML Page: Query Setup 5

Figure 3. The scheme that rules the dynamic generation of the HTML pages of the remote GUI.

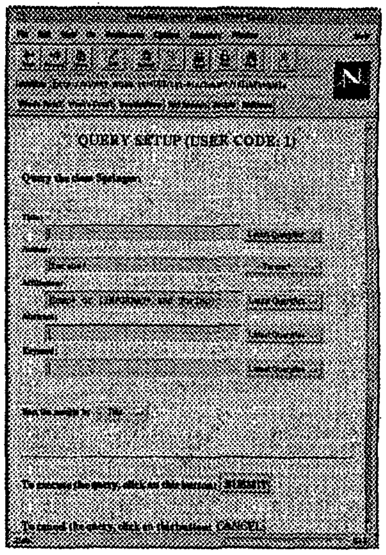

DIGITAL LIBRARY QUERY SYSTEM

(a)


CLASS OF DOCUMENTS (USER CODE: 1)

SUBMIT

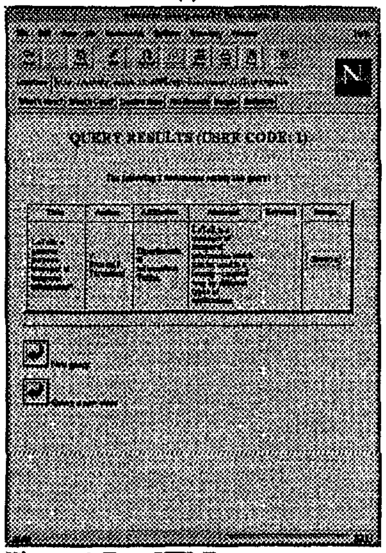
(b)


QUERY SETUP (USER CODE: 1)

SUBMIT

CANCEL

(c)


QUERY RESULTS (USER CODE: 1)

(d)

Figure 4. Four HTML pages, corresponding to types: (a) 1; (b) 4; (c) 5; (d) 6; in Fig. 3.

IEEE Computer Society.

[2] Esposito, F., D. Malerba, and G. Semeraro (1993). Automated acquisition of rules for document understanding. *Proc. of the 2nd Int'l Conf. on Document Analysis and Recognition*, 650-654. Los Alamitos: IEEE Computer Society.

[3] Esposito, F., D. Malerba, and G. Semeraro (1994). Multistrategy learning for document recognition. *Applied 'Artificial Intelligence: An Int'l Journal*, 1994, vol. 8, n. 1, 33-84.

[4] Esposito, F., D. Malerba, and G. Semeraro (1995). A Knowledge-Based Approach to the Layout Analysis. *Proc. 3rd Int'l Conf. on Document Analysis and Recognition*, 466-471. Los Alamitos: IEEE Computer Society.

[5] Esposito, F., A. Lanza, D. Malerba, and G. Semeraro (1997). An Intelligent Advisory System for Environmental Planning: Machine Learning for Map Interpretation. *Applied Artificial Intelligence: An Int'l Journal*, 1997 (to appear).

[6] Fox, E.A. (1994). How to make intelligent digital libraries. *Methodologies for Intelligent Systems - Lecture Notes in AI 869*, 27-38. Z.W. Ras and M. Zemankova (Eds.), Berlin: Springer-Verlag.

[7] Gladney, H.M., Z. Ahmed, R. Ashany, N.J. Belkin, E.A. Fox, and M. Zemankova (1994). Digital library: Gross structure and requirements. *Proc. Workshop on On-line Access to Digital Libraries*, Los Alamitos: IEEE Computer Society.

[8] Lesk, M. (1993). The Digital Library: What is it? Why should it be here? *Source Book on Digital Libraries*, Tech. Rep. TR 93-35, Virginia Tech, Dept. of Computer Science, Blacksburg, VA. Edited Volume (Ed. E.A. Fox).

[9] Malerba, D., G. Semeraro, and E. Bellisari, (1995). LEX: A knowledge-based system for the layout analysis. *Proc. 3rd Int'l Conf. on the Practical Application of Prolog*, 429-443.

[10] Semeraro, G., F. Esposito, and D. Malerba (1994). Learning contextual rules for document understanding. *Proc. 10th Conf. on Artificial Intelligence for Applications*, 108-115. Los Alamitos: IEEE Computer Society.

[11] Semeraro, G., F. Esposito, N. Fanizzi, and D. Malerba (1995). Revision of logical theories. *Topics in Artificial Intelligence - Lecture Notes in AI 992*, 27-38. M. Gori and G. Soda (Eds.), Berlin: Springer.

[12] Sievert, M.C., E.J. Mckinin, E.D. Johnson, and J.A. Mitchell (1991). Retrieval from full-text medical literature: The dream and the reality. *Proc. 15th Symp. on Computer Applications in Medical Care*, 348-352.

[13] Tang, Y.Y., C.D. Yan, and C.Y. Suen (1994). Document processing for automatic knowledge acquisition. *IEEE Trans. on Knowledge and Data Engineering*, 1994, vol. 6, no. 1, 3-21.

[14] Tsujimoto, S., and H. Asada (1990). Understanding multi-article documents. *Proc. 10th Int. Conf. on Pattern Recognition*, pp. 551-556, June 16-21, Atlantic City.

[15] Yamamoto, K. (1993). Conference Chair's Message. *Proc. 2nd Int'l Conf. on Document Analysis and Recognition*, v-v. Los Alamitos: IEEE Computer Society.