Chapter 16
FOL Learning for Knowledge Discovery in Documents

Stefano Ferilli
Università degli Studi di Bari, Italy

Floriana Esposito
Università degli Studi di Bari, Italy

Marenglen Biba
Università degli Studi di Bari, Italy

Teresa M.A. Basile
Università degli Studi di Bari, Italy

Nicola Di Mauro
Università degli Studi di Bari, Italy

ABSTRACT

This chapter proposes the application of machine learning techniques, based on first-order logic as a representation language, to the real-world application domain of document processing. First, the tasks and problems involved in document processing are presented, along with the prototypical system DOMINUS and its architecture, whose components are aimed at facing these issues. Then, a closer look is provided for the learning component of the system, and the two sub-systems that are in charge of performing supervised and unsupervised learning as a support to the system performance. Finally, some experiments are reported that assess the quality of the learning performance. This is intended to prove to researchers and practitioners of the field that first-order logic learning can be a viable solution to tackle the domain complexity, and to solve problems such as incremental evolution of the document repository.

INTRODUCTION

After some years in which it was seen as an area of interest only for research, Machine Learning (ML for short) techniques and systems have started to gain progressive credit in the everyday Computer Sci-

DOI: 10.4018/978-1-60566-766-9.ch016

ence landscape, and to be used for facing several real-world problems where classical systems show their limits. When ML systems work in real-world domains, they must typically deal with features such as complexity, need for efficiency and continuous adaptation to change. Optionally, humans may need to understand the inferred models in order to check and validate them. While the numeric and statistical setting, traditionally studied in the literature, can ensure efficiency, the other requirements call for more powerful representation formalisms. First-Order Logic (FOL for short) representation and processing techniques are more suitable than attribute-value and propositional ones for dealing with complexity and providing human understandability of the inferred models; moreover, they can deal with change and adaptation as well.

This work presents a suite of symbolic FOL learning algorithms and systems, that can serve as larger system components for satisfying these requirements in accomplishing real-world tasks. Their FOL representation language allows to effectively and efficiently deal with complex domains, yielding uniform human-understandable descriptions for observations and models. The FOL learning system INTHELEX tackles all of these requirements in a supervised setting. Being based on an incremental algorithm, it can update and refine the inferred models, instead of learning new ones from scratch, in order to account for new available evidence. Interestingly, its incremental abilities are not limited to examples processing, but allow to add to the theory and handle even completely new classes as soon as their instances come up. Conversely, when new observations become available for which a target classification is not given, an unsupervised setting is needed. In such a case, another module of the suite, that implements a recently developed similarity framework for FOL (Horn clause) representations, allows to face the problem.

This chapter focuses on Document Processing and Management, a real-world application domain that is gaining increasing interest in recent years, due to the progressive digitization of information sources. It involves almost all of the above requirements: need for quick response to the users' requests, complexity due to the high variability of documents, need for the librarians of checking and validating the inferred models, continuous flow of new documents. DOMINUS is a general-purpose document management framework that is able to process documents in standard electronic formats in order to recognize the type they belong to and their significant components based on their layout structure, and to selectively extract relevant information to be used for semantic indexing and later retrieval. Possible specific applications of the framework include support for the Semantic Web on Internet documents and content-based document management in organizations and libraries. Its architecture includes a module that provides Machine Learning services that support the different tasks involved in document processing and management. The FOL techniques presented in this chapter were embedded in such a module to provide the core functionality for making the framework powerful and flexible enough to deal with real-world cases.

In the following, after presenting the tasks and problems involved in Digital Libraries management, and how they have been tackled in DOMINUS, a more technical presentation of the incremental FOL framework exploited for document classification and understanding will be provided. The basics of the first-order logic setting will be recalled, and the similarity assessment on which the framework is based will be presented. The supervised and unsupervised learning modules and their cooperation will be then discussed, followed by experiments on a real-world dataset that show how the proposed framework can successfully and effectively be exploited as a viable solution to the incremental extension of documents and document classes in a digital library.

THE DOCUMENT PROCESSING DOMAIN

Recent evolution of computer technology in the last decades has provided the basis for exploiting computers as the principal source and repository of documents, and the Internet as the principal means of distribution, exchange and access for them. The consequent shift from paper to digital support for documents provided new opportunities for their management and processing activities, solving the problems of duplication and sharing that seriously affected legacy (paper) documents. However, making document production easy and cheap, it also introduced new problems, consisting in a huge amount of available documents (and in an associated decrease of their content quality) (Sankar, 2006). The well-known *information overload* problem, indeed, consists of the users' difficulty in accessing interesting information in large and heterogeneous repositories. Hence, the need to organize documents in a way that enables and makes easier, efficient and effective their retrieval and browsing based on an overview of the documents in the collection and of their relationships.

The primary criterion for searching interesting documents is by content. Hence, the documents in the repository should be grouped and organized accordingly. However, doing this manually is very expensive, and doing it automatically is very difficult due to the need of capturing document meaning (i.e., semantics). A more tractable starting point is exploiting layout similarity (i.e., syntax). The two approaches are often complementary, since documents of the same type/class usually have a similar spatial organization of their components and, conversely, different kinds of content are typically carried by documents having different layout styles. An additional, interesting and potentially useful relationship is that significant content is often placed in particular layout components, so that being able to identify the typical components of each group allows to selectively read only those components for identifying the document content. As a consequence, the ability to handle and manage documents according to layout structure similarity can be a key factor towards the ability to reach content-based management as well. Accordingly, document processing systems typically carry out two phases to identify the significant components of a digital document (Nagy, 2000). The former, called *Layout Analysis*, aims at identifying the geometrical organization of the components in a document page and at detecting relations among them, resulting in the so-called *Layout Structure*. The latter, called *Document Image Understanding*, aims at associating the proper logical role to each component, resulting in the so-called *Document Logical Structure*. Based on the assumption that the logical structure is different for different kinds of documents, Document Image Understanding can take advantage in terms of efficiency and accuracy if a particular sub-task, called *Document Image Classification*, is preliminarily carried out, aimed at identifying the document class (e.g., newspaper, scientific paper, email, technical report, call for papers) before associating each significant layout components for that class to a tag that expresses its role (e.g., signature, object, title, author, abstract, footnote, etc.).

Most work in the existing literature concerns the application of intelligent techniques to identify the document layout. In this chapter we focus on the identification of the proper layout class of a document, and of the main layout components of interest in it. Manual processing of the documents in the collection is clearly infeasible, not only, as already pointed out, for economic reasons, but also in principle, because of the huge amount of documents to be processed and, more seriously, due to the continuing extension of the document base which is typical in dynamic environments such as real-world Digital Libraries. This motivates the research for efficient and effective automatic techniques for document classification and understanding.

A first characteristic of the document domain is the wide variety of existing layout styles. This requires a powerful and flexible representation and processing framework, that allows to express and compare documents with different number of components, related to each other in many different ways. Attribute-Value representations can describe a document by means of a fixed set of features, each of which takes a value from a corresponding pre-specified value set. But one cannot know in advance how many components make up a generic document, and classes of documents are characterized by the presence of peculiar components and by the specific way their mutual positions and alignments are organized, rather than by their size and absolute position, that may vary even significantly. Some examples: the length of the title block could range from 1 to 4 lines, which would result in very different position of other components (such as authors and abstract) in the page, making their range of placement very wide consequently; in a scientific paper, it is useful to know that the acknowledgements usually appear above the references section and in the end of the document, or that the affiliation of the authors is reported generally at the beginning of the document, below or on the right of their names. Thus, document layout classes are hardly captured by static models, and classical attribute-value formalisms are not suitable for this task. Conversely, relational representations can capture the page layout complexity, allowing to identify invariants that characterize document classes, and additionally producing human-readable descriptions that can be exploited by domain experts for validation or understanding purposes. For these reasons, we propose to apply incremental FOL ML techniques along these phases of document processing where the classical statistical and numerical approaches to classification and learning may fail, being not able to deal with the lack of a strict layout regularity in the variety of documents available on-line.

Another issue to be considered when dealing with the document domain is the continuous flow of new and different documents in a Web repository or Digital Library. This means that any system working in this environment must be able to deal with changes in the domain, the context, or the user needs (in a word, it must be *incremental*). Traditional ML methods work according to the so-called batch approach, that requires the set of training examples, belonging to a defined number of classes, to be entirely available since the beginning, and exploits them altogether to produce a model. Thus, they assume that the classes are known and fixed since the beginning as well. If any change happens, they must restart the entire learning process to produce a model capable of coping with the new scenario. This prevents the opportunity of dealing with new instances, possibly belonging to new classes, that the learned theory does not account for. Conversely, the ability to revise a domain theory instead of restarting from scratch would be more appropriate to enable continuous responsiveness to the changes in the context, thus dealing with concept evolution, and could significantly improve efficiency. The incremental setting implicitly assumes that the information (observations) gained at any given moment is incomplete, and thus that any learned theory could be susceptible of changes. Hence incremental learning, as opposed to classical batch one, is needed whenever either incomplete information is available at the time of initial theory generation, or the nature (and the kinds) of the concepts evolves dynamically. In the document processing scenario, this would happen when the typing style of documents in a given class changes in time or when a brand new class is considered in the document repository.

RELATED WORK

Much of the work that has been done on document image analysis refers to algorithms and techniques that are applied to images of documents in order to obtain a computer-readable description from a scanned

document (Nagy, 2000). Even recently, the work on document analysis involves techniques that operate on image documents and a variety of statistical, rule-based, grammar-based techniques have been proposed as briefly reported in the following.

In (van Beusekom, 2006; van Beusekom, 2007) a new class of distance measures for document layouts is defined, based on a two-step procedure: after computing the distances between the blocks of two layouts, the blocks of one layout are assigned to the blocks of the other layout in a matching step. However, for layout structure discovering well-known layout algorithms from the literature are exploited, such as Voronoi (Kise 1998; Lu 2005), XY-Cut (Nagy, 1992), Run Length Smearing Algorithm (Wong, 1982), Docstrum (O'Gorman, 1993), Whitespace (Baird, 1994; Breuel, 2000). Sometimes, ideas in classical algorithms are enhanced with techniques for improving parameter setting, as in (Shi, 2005).

In (Marinai, 2006) a system for the retrieval of documents on the basis of layout similarity of document images is described. The system extracts the layout regions and represents them as an XY tree (Nagy, 1992). The indexing method combines a tree clustering algorithm based on the Self Organizing Maps (SOM - a particular type of artificial neural network that computes, during learning, an unsupervised clustering of the input data) with Principal Component Analysis. In such a way, similar pages are retrieved without comparing directly the query page to each indexed document. SOMs are applied in document image processing also for layout clustering aimed at document retrieval and page classification (Marinai, 2008). Other works exploit Artificial Neural Networks in document layout analysis and recognition as showed in (Marinai, 2005) or for logical document structure extraction, as in (Rangoni, 2006) extended in (Belaid, 2008). The document structure is described along a layer architecture. Each layer represents successive steps of the interpretation decomposition from physical (input) to logical (output) level of the document. The recognition process proceeds by repetitive perceptive cycles propagating information through the layers.

Other techniques exploit statistical approaches. In (Laven, 2005) a statistical pattern recognition approach to the problem of physical and logical layout analysis is investigated. The authors proposed an algorithm based on a logistic regression classifier working on a set of manually segmented and labelled page images, followed by statistical classifiers for the logical layout analysis. A similar approach can be found in (Carmagnac, 2004a) in which a semi-supervised document image classification system is presented. Given a set of unknown document images, the system uses unsupervised clustering to obtain grouping hypotheses for the same physical layout images. Then the operator can correct or validate them according to an objective, e.g. to have homogeneous groups of images whose descriptions are used for the training of the supervised document image classifier (Carmagnac, 2004) that is able to find the class of unknown documents. Another approach based on multiple hypotheses is proposed by (Kagehiro, 2008), where different hypotheses for layout segmentation, generated by low-level image analysis, are exploited by a final analysis aimed at character recognition.

A stream of works has concerned layout analysis of office documents, such as invoices, exploiting rule-based methods (Dengel, 2007), Case-Based Reasoning (Hamza, 2007) or formalized descriptions that can be compiled into executable code (Tuganbaev, 2005). Others have focused on collections of heterogeneous documents (Chen, 2007).

However, few works have faced the problem of discovering the layout and logical structure of documents in electronic formats, as opposed (but complementary) to document image analysis. (Seki, 2007) analyses document structure for simultaneous management of information in documents from various formats (image, PDF, and HTML). Most other contributions aim at extracting (some part of) the document content by means of a syntactic parsing of the PDF (Futrelle, 2003; Chao, 2003; Ramel, 2003) or

at discovering the background by means of statistical analysis applied to the numerical features of the documents and its components (Chao, 2004).

Other approaches based on domain knowledge are reported in (Hadjar, 2004; Rigamonti, 2005), where an expert provides a model for each class of documents to be handled. Such a model is represented by means of a grammar, i.e. a hierarchy of logical entities. Thus, after a step of syntactic parsing of the PDF document, aimed at discovering document primitives such as text entities, images and graphics, and after grouping the homogeneous entities discovered, the logical layout structure is recognized by labeling text entities (e.g., title, author, body) projecting them in the provided document model. A similar approach which uses grammars to annotate document components is proposed in (Anjewierden, 2001). Here, based on the model provided by an expert, a set of possible roles is assigned to each layout object. Then, they are collected into more complex objects until the logical structure is produced.

Processing of digital documents not having a strict layout is also faced by other works, such as Web pages (Feng, 2005; Burget, 2007; Guo, 2007) or e-mail messages (Chao, 2005).

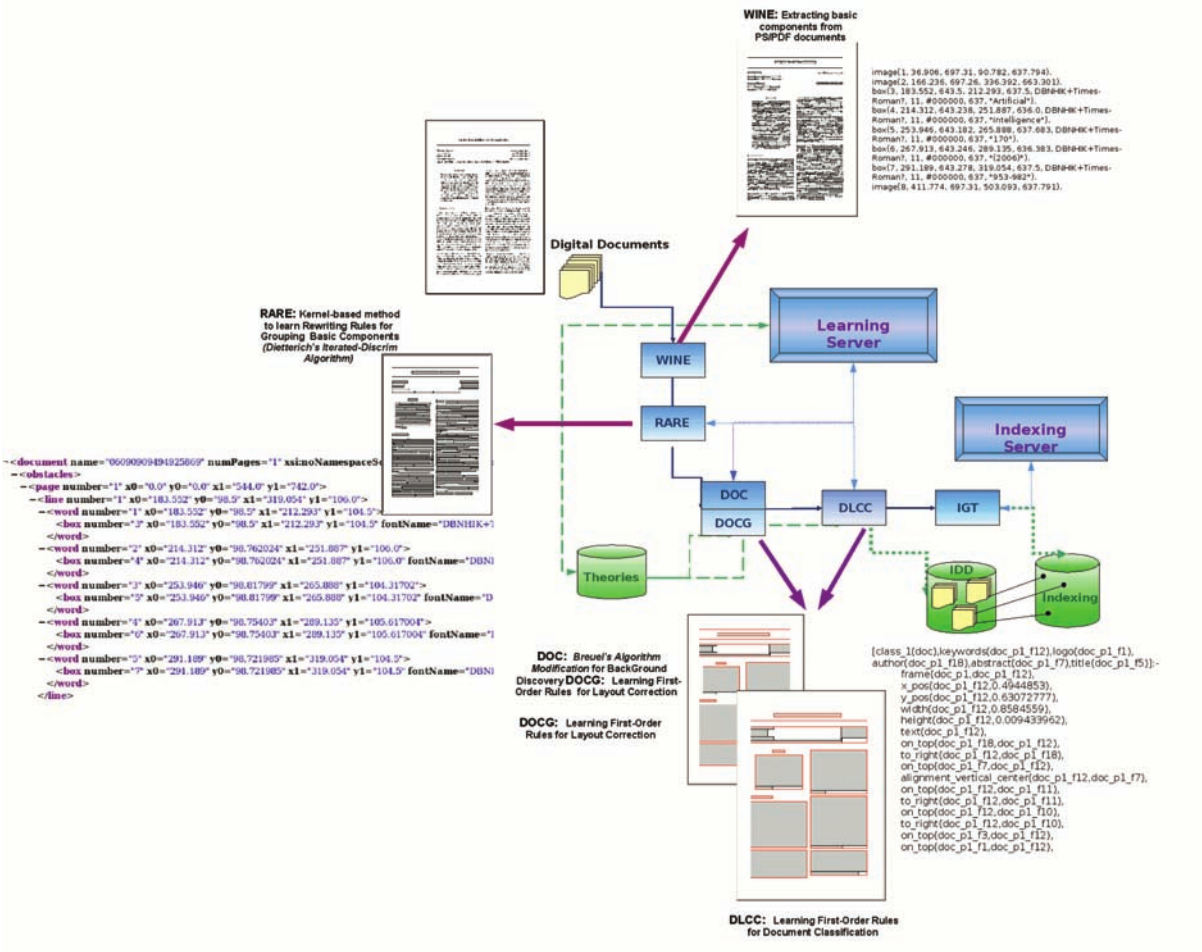
THE DOMINUS FRAMEWORK

DOMINUS (DOcument Management INtelligent Universal System) is a document processing framework characterized by the intensive exploitation of intelligent techniques in all involved tasks, from acquisition to analysis, indexing, categorization, storing and retrieval (Esposito, 2008). Its architecture, reported in Figure 1 along with the document processing flow, is general and flexible, so that it can be embedded as a document management engine into different Digital Library systems. A central role in DOMINUS is played by the Learning Server, which intervenes during different processing steps in order to continuously adapt the acquired knowledge taking into consideration new experimental evidence and changes in the context. The models inferred by the different components embedded in the Learning Server are stored for future exploitation in the Theories knowledge base.

The document layout analysis process starts with the application of a pre-processing module, called *WINE* (Wrapper for the Interpretation of Non-uniform Electronic document formats), that takes as input a digital document, translates it in an intermediate vector format that is independent on the original document format, and then produces the document's XML basic representation. Such a representation describes the document as a set of pages made up of basic blocks, often corresponding to graphical lines or rectangles, images and (groups of) text characters. Due to the large number of basic blocks, they are preliminarily aggregated in a bottom-up fashion: first, based on overlapping or adjacency criteria, composite blocks corresponding to whole words are produced; then, a further aggregation of words into lines is performed, based on inter-word spacing size. Using a fixed threshold (e.g., the mean distance between blocks) for this aggregation could be ineffective on multi-column documents, possibly merging into a single line words belonging to co-linear lines in adjacent columns. This problem was tackled through ML techniques, by casting the task to a Multiple Instance Problem and solving it by means of the kernel-based method proposed in (Dietterich, 1997). Such a method was embedded in the Learning Server, and exploited to generate rewriting rules for setting some parameters that are able to properly group word blocks into lines and will be exploited by *RARE* (Rule Aggregation REwriter).

After the line-blocks have been identified and represented by extending accordingly the XML description of the document, layout analysis proceeds top-down for collecting the semantically related blocks into consistent frames. The page background is identified first, and then the frames that surround docu-

Figure 1. DOMINUS Architecture and Processing Flow



ment content are derived as the complement of such a background. The *DOC* (Document Organization Composer) module, in charge of accomplishing this task, carries out the background structure analysis according to a modification of the algorithm proposed in (Breuel, 2002) to find iteratively the maximal white rectangles in a page. The original algorithm had to be modified in order to improve its efficiency and to stop when the significant frames are isolated, avoiding the retrieval of insignificant background white spaces such as inter-word or inter-line ones. Due to the great variability in layout both among and within documents, an all-purpose step where forcing termination of the process before its natural conclusion cannot be identified *a priori*. Rather, DOC applies some heuristics to decide, at each loop, whether exiting or continuing. Nevertheless, due to the background rectangles size and distribution, it can happen that the best stop point yields a final layout that is not ideal, because some significant white spaces have not yet been recognized by the background analysis or, conversely, because other insignificant ones have already been retrieved. These single wrong elements must be specifically corrected, usually by a manual intervention of an expert. To avoid whenever possible the user intervention, another module called *DOCG* (Document Organization Correction Generator) was implemented, in charge of learning and applying correction rules to be automatically applied on this task. These rules are automatically

learned from previous manual corrections collected by the system during the intervention of the experts on some training documents. Since the significance of a background rectangle in order to include or exclude it from the final layout is influenced by its relative size and position with respect to the other content and background rectangles around it, a FOL learning approach is needed to infer rules for this task, and a corresponding component has been added to the Learning Server to accomplish this task.

Once the layout structure recognition step successfully terminates, the *Document Image Understanding* task must be started, in charge of recognizing the document class and of associating a semantic role to each significant frame. These two sub-tasks are strongly interrelated, since the kind of components to be expected and identified in a document depends on its class (e.g., title, authors and their affiliations, abstract and references are typical of scientific papers, while in a commercial letter one might look for the sender, object, date and signature). Hence, the document class must be identified first, in order to know which kinds of components are to be expected and located, and then each component is associated to a label that properly expresses its role in the document. This task is performed by exploiting again FOL theories, since the document class is identified according to the distribution of frames in the layout, and the role played by each frame is determined by its content and spatial relationships with respect to surrounding frames. In DOMINUS, the classification and labelling theories are automatically learned, and the *DLCC* (Document and Layout Components Classifier) module takes care of managing and handling them. In a continuously evolving domain such as a digital document repository, not only new documents, but even completely new (known or unknown) classes can be expected to show up sooner or later, and hence the ability to refine and extend an existing theory according to new evidence is a fundamental feature. Hence, when a document/component of a class already known to the system is wrongly classified, the corresponding definition can be generalized or specialized accordingly, without re-starting learning from scratch. In case the document belongs to a class not yet considered by the theory, but an expert can identify it, the set of classes can be extended automatically. In case the class is unknown, the document is put in stand-by for future automatic discovery of new classes, this way dynamically extending the set of known classes. The incremental framework according to which these modifications and extensions of the theories take place in case of failure will be examined more thoroughly in the next sections. In the following, we will present the similarity-based learning framework underlying all steps of incremental learning and embedded in the Learning Server module. Such a framework is general, and in the case of DOMINUS works on FOL descriptions of the documents layout structure by identifying the description components that are more similar and hence more likely to correspond to each other.

At the end of this step both the original document and its XML representation, now including description of words, lines and frames, and enriched with class information and components annotation, have been stored in the Internal Document Database, IDD. Further steps, that are outside the scope of this work, concern text extraction from the significant components only and application of various Natural Language Processing, Information Retrieval and Information Extraction techniques that can support the semantic access to the document content. Specifically, indexing procedures are performed by the Indexing Server, useful for an effective content-based document retrieval. The *IGT* (IndexinG of Text) module currently includes full-text indexing, Latent Semantic Indexing, Keyword Extraction: the former supports classical retrieval, the second allows to retrieve documents that are related to a query although they do not contain exactly the same terms, and the latter aims at restricting the query focus only on key terms of the document.

FIRST-ORDER LOGIC PRELIMINARIES

FOL is a powerful formalism that can overcome the typical limitations shown by propositional or attribute-value representations by being able to express relations between objects. Logic Programming (Lloyd, 1987) is a FOL-based paradigm for making logic a machinable formalism. Logic programs are made up of *Horn clauses*, i.e. logical formulae of the form $l_1 \wedge ... \wedge l_n \Rightarrow l_0$, often written in Prolog notation as l_0:– l_1, ..., l_n, to be interpreted as "l_0 (*head*) is true, provided that l_1 and ... and l_n (*body*) are all true". The l_i's are *literals*, where a literal is an *atom* (a predicate applied to terms) or its negation. A term is a constant, a variable or a function symbol applied to other terms. A predicate or function is a *k*-ary symbol (or has arity *k*) if it applies to *k* terms. Program execution is cast to finding a proof for some query expressed as a conjunction of literals. Differently from attribute-value representations, in FOL there is no fixed representation of an observation, and hence no one-to-one mapping between attributes when comparing two representations can be assumed. Thus, different portions of one description can be mapped onto different portions of another, a phenomenon known as *indeterminacy* that causes serious computational problems both when inducing models and when exploiting them for prediction. A *substitution* is a mapping from variables to terms. The classical generalization model adopted in Logic Programming is θ-subsumption, according to which a clause *C* is more general than another clause *D* ($C \leq_\theta D$) if and only if there exists a substitution θ such that, applied to terms of *C*, yields a subset of *D* ($C\theta \subseteq D$). The algorithm for finding the (unique) *least general generalization* (*lgg*) under θ-subsumption between two clauses was introduced by Plotkin (1970).

Datalog (Ceri, 1990) is a restriction of logic programming, born to deal with relational databases, in which only variables and constants are allowed as terms (thus, nesting by means of functions is not allowed). Generic clauses can be translated into Datalog ones and vice-versa (Rouveirol, 1992). The Object Identity assumption (OI for short) requires that different terms must denote different objects of the domain (it is generally assumed for constants, but the OI framework applies to variables as well). Datalog^{OI} is a restriction of Datalog in which clauses are interpreted under OI, which does not cause loss in expressive power (Semeraro, 1998). OI induces a new generalization model between clauses, θ_{OI}-subsumption, whose associated space is not a lattice (as under classical θ-subsumption), with the consequence that uniqueness of the *lgg* is not guaranteed but the clause structure is fixed since two different literals can never collapse because of their variables being bound on the same terms (which makes them more mechanizable).

Given a clause *C*, we define its *associated graph* as a Directed Acyclic Graph $G_C = (V,E)$, *stratified* (i.e., with the set of nodes partitioned) with

$$V = \{l_0\} \cup \{ l_i \mid i \in \{1,...,n\}, l_i \text{ built on } k\text{-ary predicate, } k > 1\}$$
$$E \subseteq \{(a',a'') \in V \times V \mid \text{terms}(a') \cap \text{terms}(a'') \neq \varnothing \}$$

(where terms(*f*) denotes the set of terms that appear in the sub-formula *f*) built as follows. The head is the only node at level 0 (first element of the partition). Then, each successive element of the partition is made up by adding new nodes (not yet reached by edges) that have at least one term in common with nodes in the previous level. Hence, each node in the new level is linked by an incoming edge to each node in the previous level having among its arguments at least one term in common with it. The head literal, being unique, is exploited as a starting point in the graph (a kind of root) and provides a unique and well-defined perspective on the clause structure among the many possible, and hence significantly

reduces indeterminacy. For our purposes, there is no loss of generality in restricting to linked clauses (i.e., clauses whose associated graph is connected (Lloyd, 1987)): indeed, linked sub-parts of non-linked clauses can be dealt with separately, having no connection between each other.

SIMILARITY FRAMEWORK

Logic formalisms typically work on truth values, and thus only allow for binary (true/false) answers on whether a given (more general) formula accounts for another (more specific) one. Hence, the need for mechanisms to assess a degree of similarity among logic descriptions. Previous works on similarity/distance measures and techniques developed for comparing first-order descriptions are concerned with flexible matching (Esposito, 1992), supervised learning (Bisson, 1992a; Emde, 1996; Domingos, 1995; Sebag, 1997; Nienhuys-Cheng, 1998; Ramon, 2002; Kodratoff, 1986) and unsupervised learning (Thompson, 1989; Ramon, 1999; Bisson, 1992b; Blockeel, 1998). The similarity framework for FOL descriptions presented in the following overcomes some problems that are present in those works: it does not require assumptions and simplifying hypotheses (statistical independence, mutual exclusion) to ease the probability handling, no prior knowledge of the representation language is required and is not based on the presence of 'mandatory' relations, the user must not set weights on the predicates' importance, it can be easily extended to handle negative information, it avoids the propagation of similarity between sub-components that poses the problem of indeterminacy in associations, it yields a unique value as a result of a comparison, which is more understandable and comfortable for handling, it is based directly on the structure, and not on derived features.

Similarity Function

The first step in setting up a similarity framework is designing a similarity function to evaluate the similarity between two items *i'* and *i"* based on the presence of common and different features among them (Lin, 1998), that can be expressed by the following parameters:

n, the number of features owned by *i'* but not by *i"* (*residual* of *i'* wrt *i"*);
l, the number of features owned both by *i'* and by *i"*;
m, the number of features owned by *i"* but not by *i'* (*residual* of *i"* wrt *i'*).

Intuitively, a larger *l* should increase the similarity value, while larger values of *n* and *m* should decrease it. A classical formula in the literature based on these parameters is that by Tverski (1977), but its behaviour in extreme cases (full similarity or no overlapping between items features) can be problematic, for which reason we developed a novel similarity function based on the above parameters and on an additional parameter α ($0 \leq \alpha \leq 1$) that weights the importance of either item:

$$sf(\alpha, i', i'') = sf(\alpha, n, l, m) = (l+1)\,(\alpha/(l+n+2) + (1-\alpha)/(l+m+2)) \in]0,1[$$

(in the following, we will just write $sf(i', i'')$ and $sf(n, l, m)$, implicitly assuming $\alpha = 0.5$). Since FOL formulae are complex expressions, evaluating the above parameters in a comparison is not straightforward. Hence we propose to evaluate those parameters for progressively large subcomponents of FOL

formulae: for basic components the similarity is computed directly from the above formula, whereas for complex components the similarity comes from a composition of the formula applied on their direct features with the similarity of their lower level sub-components involved. In FOL formulae, terms represent specific objects, that are related to each other by means of predicates. Accordingly, two levels of similarity can be defined for pairs of first-order descriptions: the *object* level, concerning similarities terms in the descriptions, and the *structure* one, referring to how the nets of relationships in the descriptions overlap.

Object Similarity

Let us consider two clauses *C'* and *C"*. When comparing a pair of terms *a'* taken from *C'* and *a"* taken from *C"*, two kinds of object features can be distinguished: the properties they own (*characteristic features*), usually expressed by unary predicates (e.g. *female(X)*), and the roles they play (*relational features*), generally expressed by the position the object holds among *n*-ary predicate arguments (indeed, different positions refer to different roles played by the objects: e.g., in *mother(X,Y)* the first argument position identifies the role of the mother and the second one represents the role of the child).

If *P'* and *P"* are the sets of characteristic features of *a'* and *a"*, respectively, the *characteristic similarity* between *a'* and *a"* is computed as $sf_c(a',a") = sf(n_c, l_c, m_c)$ for the following parameters:

$n_c = |P' \setminus P"|$ number of properties owned by *a'* in *C'* but not by *a"* in *C"* ;
$l_c = |P' \cap P"|$ number of common properties between *a'* in *C'* and *a"* in *C"*;
$m_c = |P" \setminus P'|$ number of properties owned by *a"* in *C"* but not by *a'* in *C'* .

Similarly, if *R'* and *R"* are the multisets (indeed, one object can play many times the same role in different atoms: e.g., a mother of many children) of relational features of *a'* and *a"*, respectively, the *relational similarity* between *a'* and *a"* is computed as $sf_r(a', a") = sf(n_r, l_r, m_r)$ for the following parameters:

$n_r = |R' \setminus R"|$ how many times *a'* plays in *C'* role(s) that *a"* does not play in *C"*;
$l_r = |R' \cap R"|$ number of times that both *a'* in *C'* and *a"* in *C"* play the same role(s);
$m_r = |R" \setminus R'|$ how many times *a"* plays in *C"* role(s) that *a'* does not play in *C'*.

Overall, the *object similarity* between two terms is defined as

$sf_o(a',a") = sf_c(a',a") + sf_c(a',a") \in]0,2[$.

Structural Similarity

When checking for the structural similarity of two formulae, many objects can be involved, and hence *n*-ary atoms expressing their mutual relationships represent a constraint on how each of them in the former formula can be mapped onto another in the latter. This is the most difficult part, since relations are specific to the first-order setting and are the cause of indeterminacy. Since we want to compare any two (sub-)formulae, in the following we will consider term *associations* as an extension of substitutions

that map terms onto terms, and call *compatible* two FOL (sub-)formulae that can be mapped onto each other without yielding inconsistent term associations (i.e., a term in one formula cannot be mapped onto different terms in the other formula).

Given an *n*-ary atom, we define its *star* as the multiset of *n*-ary predicates corresponding to the atoms linked to it by some common term (indeed, a predicate can appear in multiple instantiations among these atoms). Intuitively, the star of an atom depicts 'in breadth' how it relates to the rest of the formula. The *star similarity* between two compatible *n*-ary atoms *l'* and *l"* having stars *S'* and *S"*, respectively, is computed based on the number of common and different elements in each of the two stars, represented by the following parameters:

$n_s = |S' \setminus S''|$ how many more relations *l'* has in *C'* than *l"* has in *C"*;
$l_s = |S' \cap S''|$ number of common relations between *l'* in *C'* and *l"* in *C"*;
$m_s = |S'' \setminus S'|$ how many more relations *l"* has in *C"* than *l'* has in *C'*.

Since atoms include terms as arguments, the object similarity of the involved terms must be considered as well, for all pairs of terms included in the association θ that map *l'* onto *l"* of their arguments in corresponding positions:

$$sf_s(l',l'') = sf(n_s, l_s, m_s) + Avg(\{sf_o(t',t'')\}_{t'/t'' \in \theta}) \in]0,3[.$$

Moreover, we can consider all possible paths starting from the head and reaching *leaf* nodes (those with no outcoming edges) in the graph associated to the clauses: being such paths uniquely determined gives a leverage for significantly reducing the amount of indeterminacy in the comparison. Intuitively, a path in a clause depicts 'in depth' a given portion of the relations it describes. Paths can be interpreted as the basic components of the clause structure, and exploited instead of single atoms when checking similarity between clauses.

Given two paths $p'=<l'_0,l'_1,...,l'_{n'}>$ in $G_{C'}$ and $p''=<l''_0,l''_1,...,l''_{n''}>$ in $G_{C''}$, their *intersection* is defined as the pair of longest compatible initial subsequences of *p'* and *p"*, excluding the head:

$$p' \cap p'' = (<l'_1,...,l'_k>,<l''_1,...,l''_k>) \text{ s.t. } \forall\, i = 1,...,k: l'_0,l'_1,...,l'_i \text{ compatible with } l''_0,l''_1,...,l''_i \wedge (k = n' \vee k = n'' \vee l'_0,l'_1,...,l'_{k+1} \text{ incompatible with } l''_0,l''_1,...,l''_{k+1})$$

Hence, the *path similarity* between *p'* and *p"* is computed according to the following parameters:

$n_p = n' - k$ length of the trail incompatible sequence of *p'* wrt *p"*;
$l_p = k$ length of the maximum compatible initial sequence of *p'* and *p"*;
$m_p = n'' - k$ length of the trail incompatible sequence of *p"* wrt *p'*

by considering also the star similarity values for all pairs of atoms associated by the initial compatible sequences:

$$sf_p(p',p'') = sf(n_p,l_p,m_p) + Avg(\{sf_s(l'_i,l''_i)\}_{i=1,...,k}) \in]0,4[.$$

Clause Similarity

Clause similarity is to be interpreted as follows: given two clauses, whose head literals contain the same number of arguments, the similarity between these two tuples of terms is computed as the similarity between the two bodies describing them. As for stars and paths, to assess the overall similarity between two clause bodies we need to evaluate the three parameters both globally and specifically for each body component, and then to merge the outcomes in a unique formula. As to the global parameters, they can be applied both to the number of common and different literals in the two clauses and to the number of common and different terms between them. To define what "common" means in this case, we exploit the concept of least general generalization (lgg), i.e. the most specific model for the given pair of descriptions. After computing the lgg, this is considered the common part in which counting the number of common literals and objects, while the literals and objects not covered by the lgg will be the residuals. Then, the similarity between each pair of literals of the two clauses that correspond to the same literal in the generalization can be taken into account as well.

More formally, given two clauses *C'* and *C"*, call $C = \{l_1,...,l_k\}$ their lgg, and consider the substitutions θ' and θ'' such that $\forall i = 1,...,k$: $l_i\theta' = l'_i \in C'$ and $l_i\theta'' = l''_i \in C''$, respectively. Thus, the number of common and different atoms between the two clause is as follows:

$n = |C'| - |C|$ how many atoms in *C'* are not covered by its least general generalization with respect to *C"* ;
$l = |C| = k$ maximal number of atoms that can be put in correspondence between *C'* and *C"* according to their least general generalization;
$m = |C''| - |C|$ how many atoms in *C"* are not covered by its least general generalization with respect to *C'*.

and the number of common and different objects is:

$n_o = |\text{terms}(C')| - |\text{terms}(C)|$ how many terms in *C'* are not associated by its least general generalization to terms in *C"*;
$l_o = |\text{terms}(C)|$ maximal number of terms that can be put in correspondence in *C'* and *C"* as associated by their least general generalization;
$m_o = |\text{terms}(C'')| - |\text{terms}(C))|$ how many terms in *C"* are not associated by its least general generalization to terms in *C'*.

Hence, the overall similarity between *C'* and *C"*, can be computed according to a function called *formulae similitudo* and denoted fs, by considering also the star similarity values for all pairs of atoms associated by the least general generalization, as expressed by the following formula:

$$fs(C',C'') = sf(n,l,m) \cdot sf(n_o,l_o,m_o) + Avg(\{sf_s(l'_i,l''_i)\}_{i=1,...,k}) \in \,]0,3[$$

that can obviously be normalized to]0,1[if needed. This function evaluates the similarity of two clauses according to the composite similarity of a maximal subset of their atoms that can be put in correspon-

dence (which includes both structural and object similarity), smoothed by adding the overall similarity in the number of atoms and objects between the two.

Similarity-Based Generalization

Implementing the theoretical *least general generalization* under θ_{OI}-subsumption would be NP-hard, and hence too much computationally expensive. Thus, one might prefer to have a good approximation in acceptable time. To find such an approximation, the similarity between clause paths, as defined in previous sections, can be exploited, in order to quickly locate subcomponents of the clauses to be generalized that best match to each other. The algorithm, presented in (Ferilli, 2007) and reported below, works as follows: generalizations of all couples of paths in the original clauses to be generalized are scanned by decreasing similarity and added to the current partial generalization if compatible with it, or ignored otherwise. When all path generalizations have been considered, a generalization is obtained. Here, the former clause is taken as a pattern for the generalization, but in the end constants in such a pattern must be replaced with new variables (not yet appearing in the generalization, a different variable for each constant) to get the actual generalization. Although this is a typical greedy approach, backtracking can be performed to get more generalizations.

```
function lgg_OI(E, C: clause): clause;
P_C ← paths(C); P_E ← paths(E);
P ← {(p_C, p_E) ∈ P_C × P_E | p_C ∩ p_E ≠ (< >,< >)};
G ← ∅; θ ← ∅
while (P ≠ ∅)
   (p_C, p_E) ← argmax_(pC,pE) ∈ P(sf(p_C, p_E))
   P ← P \ {(p_C, p_E)}
   (g_C, g_E) ← p_C ∩ p_E
   θ_q ← mapping between terms s.t. g_C θ_q = g_E if (θ_q compatible with θ)
      G ← G ∪ g_C; θ ← θ ∪ θ_q
   G ← replace all constants in G with new variables
return G
```

INCREMENTAL LEARNING FRAMEWORK

The proposed general incremental learning framework works as follows. Initially, a theory is provided, according to which classifying new observations that the system must process. Such a theory can be provided by an expert (the logical setting allows an easy formalization by humans thanks to its understandability), or have been previously learned by a ML system from examples, or even be empty (in which case the system must learn it instead of simply revising it). When the classifier is not able to recognize new incoming observations (which can be recognized because of no classification given, or multiple inconsistent classification, or a classification confidence under a given threshold, or a direct intervention of an expert), two cases can occur. If an expert can provide the correct classification, the system should modify accordingly the available theory. Such a modification can involve a refinement of the available definitions for known concepts, or require the addition of a brand new class in the theory

(which is a problem for all current ML systems, that require the whole set of concepts to be learned to be known since the beginning). Otherwise, if no class information can be provided for the problematic observation, it is put in stand-by until a sufficient number of rejections is reached, so that unsupervised learning can be run on them in order to automatically identify new classes according to their similarity. The *clustering* task aims at organizing a collection of unlabelled patterns into groups (clusters) of homogeneous elements based on their similarity. The similarity measure exploited to evaluate the distance between elements is responsible for the effectiveness of the clustering algorithms. Each discovered cluster becomes a new class, to be taken into account when considering new observations.

The supervised part of the above framework is carried out by INTHELEX (INcremental THEory Learner from Examples), a learning system for the induction of hierarchical theories from positive and negative examples. It is fully and inherently incremental: the initial theory can be empty and examples are considered and processed one by one according to a *closed loop* process, where feedback on performance (incorrectness of the theory on a new example) is used to activate the theory revision phase (Becker, 1985) in order to restore completeness and consistency. Theories learned by INTHELEX are Prolog programs implementing a classifier for the learned concepts. Examples in INTHELEX are definite ground Horn clauses, whose body describes the observation by means of only basic non-negated predicates of the representation language adopted for the problem at hand, and whose head reports the target class of a (tuple of) object(s) in the observation. In case of a negative example, the head is negated. A single observation may stand as an example or a counterexample for many concepts: a positive example for a concept is not considered as a negative example for all the other concepts (unless it is explicitly stated). A historical memory of all processed examples guarantees correctness of the future versions of the theory on the whole set of known examples.

INTHELEX is endowed with multiple inference strategies, according to the Inferential Theory of Learning framework (Michalski, 1994), in order to improve effectiveness and efficiency of the learning task. *Deduction* exploits the definitions in the learned theory and/or those in the Background Knowledge, if any, to recognize known objects in an example description. A dependency graph describes which concepts can contribute to the definition of which others. Whenever a new example is taken into account, its description is saturated with all instances of its sub-concepts in the dependency graph that can be recognized in its description. *Abstraction* allows to describe examples in a higher-level language to facilitate the generation of rules. In the abstraction framework adopted by INTHELEX (Zucker, 1998), abstraction operators defined in an Abstraction Theory, if any, can eliminate superfluous details, group specific component patterns into compound objects, reduce the number of object attributes, ignore the number of instances of a given object and obtain a coarser grain-size for attribute values. *Abduction* consists in hypothesizing unknown facts to be added to an observation, provided they are consistent with given integrity constraints, in such a way that the examples they represent are explained (if positive) or rejected (if negative) without modifying the current theory. INTHELEX adopts the abductive procedure by Kakas and Mancarella (Kakas, 1990), adapted to cope with OI.

As to *induction*, INTHELEX can learn simultaneously various concepts, possibly related to each other. When a new example is not correctly classified by the current theory, it is exploited by the *refinement operators* to fix the incorrect hypotheses. In particular, when a positive example is not covered, the theory can be revised in one of the following ways (listed by decreasing priority) such that completeness is restored:

- replacing a clause in the theory with one of its generalizations against the problematic example;
- adding a new clause to the theory, obtained by properly turning constants into variables in the problematic example (such a clause can even refer to a brand new concept);
- adding the problematic example as a positive exception.

When a negative example is covered, the theory consistency can be restored by performing one of the following actions (by decreasing priority):

- adding positive literals that characterize all the past positive examples of the concept (and exclude the problematic one) to the clauses that concur to the example coverage (starting from the lowest possible level);
- adding a negative literal that is able to discriminate the problematic example from all the past positive ones to the top-level clause in the theory by which the problematic example is covered;
- adding the problematic example as a negative exception.

Thus, examples are never rejected. Moreover, it does not need to know in advance what is the whole set of concepts to be learned: it learns a new concept as soon as examples about it are available. Thus, the algorithm implemented in INTHELEX allows to deal with two cases of incremental refinement of a theory: modification of the available hypotheses for known concepts (either extending or restricting their domains), and addition of hypotheses concerning new concepts not yet considered by the theory.

As to the unsupervised part of the framework, the proposed distance measure for FOL Horn Clauses can be exploited by any of the clustering techniques developed in the literature (Jain, 1999). As cluster prototypes, *medoid* must be exploited instead of centroids. The medoid of a cluster is defined as the observation in the cluster that has the minimum average distance from all the other members of the cluster, and is needed in a relational setting since first-order logic formulae do not induce an Euclidean space. Once the new clusters/class have been induced, three possibilities are available to assign new observations to one of them:

- such classes can be incorporated in the old theory, exploiting again INTHELEX to learn definitions for each of them according to the corresponding observations (*Conceptual Clustering*);
- otherwise, a *k*-Nearest Neighbour classification technique can be exploited, based on the same distance measure, in order to assign the new observation to the majority class among the closest instances;
- the new observation can be associated to the nearest medoid in the discovered classes.

EXPERIMENTS

The proposed strategy has been applied to real-world cases of document collections managed by DOMINUS. The system was run under Microsoft WindowsXP Professional™ on a PC endowed with a 2.13 GHz Intel processor and 2GB RAM. The dataset consisted in 353 scientific papers in PostScript or Portable Document Format, whose first pages layout descriptions were automatically generated by DOMINUS. According to these descriptions, the objective was identifying the papers' series and significant components. The documents belong to 4 different classes: Elsevier journals, Springer-Verlag

Lecture Notes (SVLN) series, Journal of Machine Learning Research (JMLR) and Machine Learning Journal (MLJ). Processing such a dataset involves a number of difficulties, due to several considerably complex aspects which were purposely introduced when choosing the documents to be included. First of all, the layout styles of the classes are quite similar to each other, which forces the automatic system to be able to grasp subtle differences in order to properly group them in distinct classes. Moreover, on the computational complexity side, efficiency is stressed since the 353 documents are described with a total of 67920 atoms, which yields an average of more than 192 atoms per description (however, some particularly complex layouts required more than 400 atoms). Lastly, the description language extensively exploits an inclusion relation between layout components that increases indeterminacy and hence can stress significantly the similarity computation. Runtime for the computation of the similarity between two observations in this dataset is on average of about 2sec, which is a reasonable time considering the descriptions complexity and the fact that the prototype has not been optimized in this preliminary version. A sample (short) document description is the following:

```
[not(icml(ferilli02)), svln(ferilli02), not(elsevier(ferilli02)),
not(ecai(ferilli02))]:-
   first_page(ferilli02, ferilli02_p1),
   frame(ferilli02_p1, ferilli02_p1_f5), text(ferilli02_p1_f5),
   to_right(ferilli02_p1_f1, ferilli02_p1_f5),
   on_top(ferilli02_p1_f5, ferilli02_p1_f1),
   on_top(ferilli02_p1_f12, ferilli02_p1_f5),
   to_right(ferilli02_p1_f12, ferilli02_p1_f5),
   on_top(ferilli02_p1_f3, ferilli02_p1_f5),
   to_right(ferilli02_p1_f3, ferilli02_p1_f5),
   on_top(ferilli02_p1_f2, ferilli02_p1_f5),
   to_right(ferilli02_p1_f2, ferilli02_p1_f5),
   on_top(ferilli02_p1_f4, ferilli02_p1_f5),
   to_right(ferilli02_p1_f4, ferilli02_p1_f5),
   frame(ferilli02_p1, ferilli02_p1_f12), text(ferilli02_p1_f12),
   to_right(ferilli02_p1_f1, ferilli02_p1_f12),
   on_top(ferilli02_p1_f12, ferilli02_p1_f1),
   center_vertical_aligned(ferilli02_p1_f12, ferilli02_p1_f1),
   on_top(ferilli02_p1_f3, ferilli02_p1_f12),
   center_vertical_aligned(ferilli02_p1_f12, ferilli02_p1_f3),
   to_right(ferilli02_p1_f2, ferilli02_p1_f12),
   on_top(ferilli02_p1_f12, ferilli02_p1_f2),
   center_vertical_aligned(ferilli02_p1_f12, ferilli02_p1_f2),
   to_right(ferilli02_p1_f4, ferilli02_p1_f12),
   on_top(ferilli02_p1_f12, ferilli02_p1_f4),
   center_vertical_aligned(ferilli02_p1_f12, ferilli02_p1_f4),
   frame(ferilli02_p1, ferilli02_p1_f4), text(ferilli02_p1_f4),
   on_top(ferilli02_p1_f4, ferilli02_p1_f1),
   center_vertical_aligned(ferilli02_p1_f4, ferilli02_p1_f1),
   on_top(ferilli02_p1_f3, ferilli02_p1_f4),
```

```
   to_right(ferilli02_p1_f4, ferilli02_p1_f3),
   center_vertical_aligned(ferilli02_p1_f4, ferilli02_p1_f3),
   on_top(ferilli02_p1_f4, ferilli02_p1_f2),
   to_right(ferilli02_p1_f4, ferilli02_p1_f2),
   center_vertical_aligned(ferilli02_p1_f4, ferilli02_p1_f2),
   frame(ferilli02_p1, ferilli02_p1_f1), text(ferilli02_p1_f1),
   on_top(ferilli02_p1_f3, ferilli02_p1_f1),
   to_right(ferilli02_p1_f1, ferilli02_p1_f3),
   center_vertical_aligned(ferilli02_p1_f1, ferilli02_p1_f3),
   on_top(ferilli02_p1_f2, ferilli02_p1_f1),
   to_right(ferilli02_p1_f1, ferilli02_p1_f2),
   center_vertical_aligned(ferilli02_p1_f1, ferilli02_p1_f2),
   frame(ferilli02_p1, ferilli02_p1_f2), text(ferilli02_p1_f2),
   on_top(ferilli02_p1_f3, ferilli02_p1_f2),
   to_right(ferilli02_p1_f2, ferilli02_p1_f3),
   center_vertical_aligned(ferilli02_p1_f2, ferilli02_p1_f3),
   frame(ferilli02_p1, ferilli02_p1_f3), text(ferilli02_p1_f3),
   width_medium(ferilli02_p1_f5), height_very_very_small(ferilli02_
p1_f5),
   width_medium_large(ferilli02_p1_f12), height_very_
small(ferilli02_p1_f12),
   width_medium_large(ferilli02_p1_f3), height_very_small(ferilli02_
p1_f3),
   width_large(ferilli02_p1_f2), height_medium(ferilli02_p1_f2),
   width_very_large(ferilli02_p1_f1), height_large(ferilli02_p1_f1),
   width_very_large(ferilli02_p1_f4), height_medium_small(ferilli02_
p1_f4),
   pos_left(ferilli02_p1_f5), pos_center(ferilli02_p1_f1),
   pos_center(ferilli02_p1_f12), pos_center(ferilli02_p1_f2),
   pos_center(ferilli02_p1_f3), pos_center(ferilli02_p1_f4),
   pos_upper(ferilli02_p1_f12), pos_upper(ferilli02_p1_f3),
   pos_upper(ferilli02_p1_f4), pos_middle(ferilli02_p1_f2),
   pos_middle(ferilli02_p1_f5), pos_lower(ferilli02_p1_f1).
```

where clause body contains the actual description of the *ferilli02* document's first page layout, and the head states this document belongs to class *svln* (Springer Verlag Lecture Notes series), and does not belong to the other classes ICML, Elsevier, ECAI.

As to supervised learning, it was started with an empty theory. This means that no class was initially known and all classes in the final theory result from incremental revisions. Two versions of INTHELEX were run and compared to each other: the classical one (I), and a new one endowed with the similarity-guided generalization (SF) of (Ferilli, 2007). It is interesting to note that, on the document dataset, the similarity-driven generalization was very effective, preserving on average more than 90% atoms of the shortest clause, with a maximum of 99.48% and just 0.006 variance. This confirms the ability of the similarity technique to guide the generalization, so that the theoretical least general one is strictly ap-

Table 1. Document classification results.

	JMLR		Elsevier		MLJ		SVLN	
	SF	I	SF	I	SF	I	SF	I
Time	589	1962	53	304	3213	2975	399	663
Acc	0.98	0.98	1	0.99	0.96	0.93	0.98	0.94
F1	0.97	0.97	1	0.97	0.94	0.91	0.97	0.93

proximated and often exactly caught. The experiment on learning classification theories was set up as a 10-fold cross-validation, whose results according to different parameters are reported in Table 1. The similarity-driven version outperformed the classical one in all considered parameters:

- runtime
- learning behaviour (number of alternative definitions per concept, number of exceptions, number of generalizations/specializations performed)
- predictive accuracy (ratio of correctly classified examples): (TP + TN) / (TP + FP + TN + FN)
- F-measure, a weighted average of the following two indicators (in our experiments equal weight was assigned to both, a setting known as F1-measure):
  - Precision (ratio of correctly classified positive examples to the total number of examples classified as positive): TP / (TP + FP)
  - Recall (ratio of correctly classified positive examples to the number of actual positive examples): TP / (TP + FN)

where:

- TP (True Positive): number of positive examples predicted as positive by the theory;
- FP (True Positive): number of negative examples predicted as positive by the theory;
- TN (True Positive): number of negative examples predicted as negative by the theory;
- FN (True Positive): number of positive examples predicted as negative by the theory.

In the whole experiment, the similarity-driven version saved 1.15 hours, resulting in averages of 98% for predictive accuracy and 96% for F1-measure (+1% and +2%, respectively, with respect to the classical version). The high value of F-measure, in particular, is very encouraging, since it ensures that the technique behaves equally well on both positive and negative examples, although the latter are three times the former. A supervised experiment was run also for the understanding task. Specifically, the following labels were considered significant and searched for, where applicable: Title, Abstract, Author and Keywords, Logo. Overall averages are slightly worse than the classical version: 95% for accuracy and 89% for F-measure (–1% and –2%, respectively, with respect to the old version), but with huge runtime savings (between 1/3 and 1/2 of the time in single folds, for a total amount of 76.46 hours – i.e. 3.19 days! – overall). Since the proposed similarity framework is made up of a new similarity function and a similarity composition strategy, we have also checked whether the strategy alone, or the similarity function as well, are responsible for the good performance. For this purpose, we replaced the novel function with other well-known measures in the literature (Jaccard's, Tverski's and Dice's) in the proposed

Table 2. Document clustering results

	Elsevier	SVLN	JMLR	MLJ	TOTAL
Prec (%)	80	98.46	90.48	97.46	91.60
Rec (%)	100	85.33	100	87.79	93.28
Pur (%)					92.35

strategy. The new measure produced improvements up to +5.48% for precision, up to + 8.05% for recall and up to +2.83% for accuracy, which confirmed its actual contribution to advance the state-of-the-art. Here is a sample definition learned for classification:

```
ecai(A):-
   first_page(A, B), frame(B, C), text(C), height_very_very_
small(C),
   on_top(D, C), text(D), pos_upper(D), frame(B, D).
```

to be read as "A document *A* belongs to class ECAI if in its first page *B* it contains two text frames, *C* and *D*, the latter in upper position in the page and above the former, having very very small height".

As to the unsupervised setting, a classical *K*-means algorithm based on the same similarity computation was exploited on the same dataset, by hiding each document's class to the system and asking it to identify 4 clusters (that would hopefully correspond to the actual document classes). The stop criterion was set as the moment in which the output of a new iteration is equal to a partition already seen in previous iterations (Ferilli, 2008). Specifically, the conceptual clustering option was chosen: i.e., after clustering, definitions for each cluster were learned exploiting INTHELEX on the same descriptions, tagged with the cluster label to which they were assigned. To evaluate performance, i.e. whether the clustering procedure was able to autonomously infer the correct classes, we checked the quality of the result by measuring the overlapping of each group with the correct classes (*supervised clustering*) according to precision, recall and purity (informally, this expresses the overlapping between classes and clusters and can be considered for clustering what predictive accuracy is in supervised learning). To do this, we had to decide which class each cluster had to be compared to. In fact, for each cluster the precision-recall values were neatly high for one class and considerably low for all the others, thus the choice of which was the best-matching class to be associated to each cluster became straightforward. Results, summarized in Table 2, show overall values well above 90%, near to those obtained by supervised learning, which indicates that the proposed method is highly effective in recognizing the original classes.

More in-depth analysis reveals that clusters associated to Elsevier and JMLR classes include all of the correct documents, reaching 100% recall; precision is also very high for JMLR, still high but neatly

Table 3. k-Nearest Neighbour classification results

	Elsevier	SVLN	JMLR	MLJ	TOTAL
Acc (%)	100	89.70	90.03	100	94.73

lower in the case of Elsevier. On the other hand, the quality of clusters corresponding to SVLN and MLJ are very similar among them, but opposite to the former cases, with a high precision balanced by a slightly lower recall. I any case, it should be taken into account that the layout styles of some of the classes in the dataset are very similar. Overall clustering runtime was in the order of hours, but since this task is not frequently carried out one could think, in a real Digital Library environment, to run it off-line and make the new classes available only after the procedure has been accomplished.

To complete the similarity-based framework for document image understanding, we also ran experiments on instance-based classification. Specifically, a k-NN approach was run to check its viability by comparing its performance to that of the theories learned by INTHELEX on the same dataset. *k* was set to 17, that is the square root of the number of learning instances (9/10 of the whole dataset in a 10-fold cross-validation setting, hence about 317) according to usual practice. Although we chose an odd (and prime) *k*, it should be noticed that, since the task was run on a multi-class dataset, the nearest neighbours are not bound to a binary classification and ties are possible. The average predictive accuracy results for the 10 folds on each class are reported in Table 3.

These results, very close to those obtained by learning conceptual definitions for each class, mean that few documents were associated to the wrong class, and hence are a further confirmation that the distance technique is very effective in identifying the proper similarity features within the given descriptions. Actually, very often in the correct cases not just a tiny majority, but almost all of the nearest neighbours to the description to be classified were from the same class. Errors were concentrated in the SVLN and JMLR classes, where, however, high accuracy rates were reached. MLJ seems quite distinct from the other classes, while Elsevier, although well-recognizable in itself, is somehow in between JMLR and SVLN, which are also close to each other. Interestingly, wrong classifications concerning Elsevier are unidirectional: JMLR and SVLN are sometimes confused with Elsevier, but the opposite never happened. Conversely, in the case of JMLR and SVLN it is bidirectional, suggesting a higher resemblance between the two. As to the comparison to the concept-learning algorithm, it is interesting to note that, while high performance on Elsevier and low performance on SVLN are confirmed from the conceptual learning case, for MLJ and JMLR the behaviour of the two approaches is opposite, suggesting somehow complementary advantages of each. This can be explained with the fact that printed journals impose a stricter fulfilment of layout style and standards, and hence their instance are more similar to each other. Thus, the concept learning algorithm is more suitable to learn definitions that generalize on peculiar features of such classes.

CONCLUSION

Digital libraries management is gaining increasing attention in a world in which the amount of available documents is growing so quickly that finding the needed information is harder and harder. Many tasks involved in document management can be profitably faced by Artificial Intelligence and Machine Learning techniques. This chapter specifically deals with the document image understanding, where first-order logic representation formalisms can provide the flexibility needed to manage documents with very different and variable layout, and incremental approaches are needed to deal with the continuous extension of the knowledge base by means of new documents and new document classes. A similarity-based framework for supervised and unsupervised learning is presented, as embedded in the prototypical document management system DOMINUS, that is able to refine existing class definitions according to

new evidence, but also to autonomously extend the set of known or unknown classes whenever needed, without restarting from scratch the learning process.

Experimental results on a real-world document dataset concerning scientific papers confirm that the proposed framework can efficiently and effectively face the problem, and that first-order logic representations and incremental approaches are a viable solution to document image understanding, with performance above 90% for all tasks: supervised learning, clustering and *k*-Nearest Neighbour classification. This is very encouraging, considering that precision and recall are typically contrasting parameters, and especially in the perspective of the representation-related difficulties. Future work will concern experimentation on different document types, optimization of the technique for reducing runtime and further extension of the framework with other techniques that can improve the overall behaviour, such as numerical and statistical techniques for sub-tasks that do not require the full power of first-order logic and can provide more efficiency.

REFERENCES

Anjewierden, A. (2001). AIDAS: Incremental logical structure discovery in pdf documents. In *Proceedings of the 6th International Conference on Document Analysis and Recognition* (pp. 374-378). Washington, DC: IEEE Computer Society.

Baird, H. S. (1994). Background structure in document images. In H. Bunke, P. S. P. Wang, & H. S. Baird (Eds.), *Document image analysis* (pp. 17-34). Singapore: World Scientific.

Becker, J. M. (1985). *Inductive learning of decision rules with exceptions: Methodology and experimentation*. Unpublished bachelor's dissertation, University of Illinois at Urbana-Champaign.

Belaïd, A., & Rangoni, Y. (2008). Structure extraction in printed documents using neural approaches. In S. Marinai & H. Fujisawa (Eds.), *Machine learning in document analysis and recognition* (pp. 1-24). Berlin, Germany: Springer.

Bisson, G. (1992a). Learning in FOL with a similarity measure. In W. R. Swartout (Ed.), *Proceedings of the 10th National Conference on Artificial Intelligence – AAAI-92* (pp. 82-87).

Bisson, G. (1992b). Conceptual clustering in a first order logic representation. In *Proceedings of the 10th European conference on Artificial intelligence*. New York: John Wiley & Sons.

Blockeel, H., Raedt, L. D., & Ramon, J. (1998). Top-down induction of clustering trees. In J. Shavlik (Ed.), *Proceedings of the 15th International Conference on Machine Learning*. San Francisco: Morgan Kaufmann.

Breuel, T. M. (2002). Two geometric algorithms for layout analysis. In *Proceedings of the 5th Workshop on Document Analysis Systems*.

Burget, R. (2007). Layout based information extraction from HTML documents. In *Proceedings of the 9th International Conference on Document Analysis and Recognition* (pp. 624-628). Washington, DC: IEEE Computer Society.

Carmagnac, F., Héroux, P., & Trupin, E. (2004a). Multi-view hac for semi-supervised document image classification. In S. Marinai & A. Dengel (Eds.), *Proceedings of the 6th International Workshop on Document Analysis Systems* (pp. 191-200). Berlin, Germany: Springer.

Carmagnac, F., Héroux, P., & Trupin, E. (2004b). Distance based strategy for document image classification. In *Advances in pattern recognition* (pp. 894-902). Berlin, Germany: Springer.

Ceri, S., Gottlob, G., & Tanca, L. (1990). *Logic programming and databases*. Berlin, Germany: Springer.

Chao, H. (2003). Graphics extraction in PDF document. In T. Kanungo, E. H. B. Smith, J. Hu, & P. B. Kantor (Eds.), *SPIE - The International Society for Optical Engineering. Volume 5010*, (pp. 317-325).

Chao, H., & Fan, J. (2004). Layout and content extraction for pdf documents. In S. Marinai & A. Dengel (Eds.), *Proceedings of the 6th International Workshop on Document Analysis Systems* (pp. 213-224). Berlin: Springer.

Chao, H., & Lin, X. (2005). Capturing the layout of electronic documents for reuse in variable data printing. In *Proceedings of the 8th International Conference on Document Analysis and Recognition* (pp. 940-944). Washington, DC: IEEE Computer Society.

Chen, S., Mao, S., & Thoma, G. (2007). Simultaneous layout style and logical entity recognition in a heterogeneous collection of documents. In *Proceedings of the 9th International Conference on Document Analysis and Recognition* (pp. 118-122). Washington, DC: IEEE Computer Society.

Dengel, A. (2007). Learning of pattern-based rules for document classification. In *Proceedings of the 9th International Conference on Document Analysis and Recognition* (pp. 123-127). Washington, DC: IEEE Computer Society.

Dietterich, T. G., Lathrop, R. H., & Lozano-Perez, T. (1997). Solving the multiple instance problem with axis-parallel rectangles. *Artificial Intelligence*, *89*(1-2), 31–71. doi:10.1016/S0004-3702(96)00034-3

Domingos, P. (1995). Rule induction and instance-based learning: A unified approach. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence* (pp. 1226-1232). San Francisco, CA: Morgan Kaufmann.

Emde, W., & Wettschereck, D. (1996). Relational instance based learning. In L. Saitta (Ed.), *Proceedings of the 13th International Conference on Machine Learning* (pp. 122-130).

Esposito, F., Ferilli, S., Basile, T. M. A., & Di Mauro, N. (2008). Machine learning for digital document processing: From layout analysis to metadata extraction. In S. Marinai & H. Fujisawa (Eds.), *Machine learning in document analysis and recognition* (pp. 79-112). Berlin, Germany: Springer.

Esposito, F., Malerba, D., & Semeraro, G. (1992). Classification in noisy environments using a distance measure between structural symbolic descriptions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *14*(3), 390–402. doi:10.1109/34.120333

Feng, J., Haffner, P., & Gilbert, M. (2005). A learning approach to discovering Web page semantic structures. In *Proceedings of the 8th International Conference on Document Analysis and Recognition* (pp. 1055-1059). Washington, DC: IEEE Computer Society.

Ferilli, S., Basile, T. M. A., Di Mauro, N., Biba, M., & Esposito, F. (2007). Similarity-guided clause generalization. In R. Basili, & M. T. Pazienza (Eds.), *Proceedings of the AI*IA-2007: Artificial Intelligence and Human-Oriented Computing* (pp. 278-289). Berlin, Germany: Springer.

Ferilli, S., Basile, T. M. A., Di Mauro, N., Biba, M., & Esposito, F. (2008). Generalization-based similarity for conceptual clustering. In Z. W. Ras, S. Tsumoto, & D. Zighed (Eds.), *Mining complex data* (pp. 13-26). Berlin, Germany: Springer.

Futrelle, R. P., Shao, M., Cieslik, C., & Grimes, A. E. (2003). Extraction, layout analysis and classification of diagrams in PDF documents. In *Proceedings of the 7th International Conference on Document Analysis and Recognition* (pp. 1007-1014). Washington, DC: IEEE Computer Society.

Guo, H.-F., Mahmud, J., Borodin, Y., Stent, A., & Ramakrishnan, I. V. (2007). A general approach for partitioning Web page content based on geometric and style information. In *Proceedings of the 9th International Conference on Document Analysis and Recognition* (pp. 929-933). Washington, DC: IEEE Computer Society.

Hadjar, K., Rigamonti, M., Lalanne, D., & Ingold, R. (2004). Xed: A new tool for extracting hidden structures from electronic documents. In *Proceedings of the 1st International Workshop on Document Image Analysis for Libraries* (p. 212). Washington, DC: IEEE Computer Society.

Hamza, H., Belaïd, Y., & Belaïd, A. (2007). A case-based reasoning approach for invoice structure extraction. In *Proceedings of the 9th International Conference on Document Analysis and Recognition* (pp. 327-331). Washington, DC: IEEE Computer Society.

Jain, A. K., Murty, M. N., & Flynn, P. J. (1999). Data clustering: A review. *ACM Computing Surveys*, *31*(3), 264–323. doi:10.1145/331499.331504

Kagehiro, T., & Fujisawa, H. (2008). Multiple hypotheses document analysis. In S. Marinai & H. Fujisawa (Eds.), *Machine learning in document analysis and recognition* (pp. 1-24). Berlin, Germany: Springer.

Kakas, A. C., & Mancarella, P. (1990). On the relation of truth maintenance and abduction. In *Proceedings of the 1st Pacific Rim International Conference on Artificial Intelligence*, Nagoya, Japan.

Kise, K., Sato, A., & Iwata, M. (1998). Segmentation of page images using the Area Voronoi Diagram. *Computer Vision and Image Understanding*, *70*(3), 370–382. doi:10.1006/cviu.1998.0684

Kodratoff, Y., & Ganascia, J.-G. (1986). Improving the generalization step in learning. In R. S. Michalski, J. G. Carbonell, & T. M. Mitchell (Eds.), *Machine learning: An artificial intelligence approach: Volume II* (pp. 215-244). Los Altos, CA: Kaufmann.

Laven, K., Leishman, S., & Roweis, S. T. (2005). A statistical learning approach to document image analysis. In *Proceedings of the 8th International Conference on Document Analysis and Recognition* (pp. 357-361). Washington, DC: IEEE Computer Society.

Lin, D. (1998). An information-theoretic definition of similarity. In *Proceedings of the 15th International Conference on Machine Learning* (pp. 296-304). San Francisco, CA: Morgan Kaufmann.

Lloyd, J. W. (1987). *Foundations of logic programming* (2nd ed.). New York: Springer-Verlag.

Lu, Y., & Tan, C. L. (2005). Constructing Area Voronoi Diagram in document images. In *Proceedings of the 8th International Conference on Document Analysis and Recognition* (pp. 342-346). Washington, DC: IEEE Computer Society.

Marinai, S. (2008). Self-organizing maps for clustering in document image analysis. In S. Marinai & H. Fujisawa (Eds.), *Machine learning in document analysis and recognition* (pp. 381-408). Berlin, Germany: Springer.

Marinai, S., Gori, M., & Soda, G. (2005). Artificial neural networks for document analysis and recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *27*(1), 23–35. doi:10.1109/TPAMI.2005.4

Marinai, S., Marino, E., & Soda, G. (2006). Tree clustering for layout based document image retrieval. In *Proceedings of the 2nd International Workshop on Document Image Analysis for Libraries* (pp. 243-253). Washington, DC: IEEE Computer Society.

Michalski, R. S. (1994). Inferential theory of learning. Developing foundations for multistrategy learning. In R. S. Michalski & G. Tecuci (Eds.), *Machine learning. a multistrategy approach, volume IV* (pp. 3-61). San Mateo, CA: Morgan Kaufmann.

Nagy, G. (2000). Twenty years of document image analysis in PAMI. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *22*(1), 38–62. doi:10.1109/34.824820

Nagy, G., Seth, S., & Viswanathan, M. (1992). A prototype document image analysis system for technical journals. *Computer*, *25*(7), 10–22. doi:10.1109/2.144436

Nienhuys-Cheng, S. (1998). Distances and limits on Herbrand interpretations. In D. Page (Ed.), *Proceedings of the ILP-98*. Berlin, Germany: Springer.

O'Gorman, L. (1993). The document spectrum for page layout analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *15*(11), 1162–1173. doi:10.1109/34.244677

Plotkin, G. D. (1970). A note on inductive generalization. *Machine Intelligence*, *5*, 153–163.

Ramel, J. Y., Crucianu, M., Vincent, N., & Faure, C. (2003). Detection, extraction and representation of tables. In *Proceedings of the 7th International Conference on Document Analysis and Recognition* (pp. 374-378). Washington, DC: IEEE Computer Society.

Ramon, J. (2002). *Clustering and instance based learning in first order logic*. Unpublished doctoral dissertation, K.U. Leuven, Belgium.

Ramon, J., & Dehaspe, L. (1999). Upgrading Bayesian clustering to first order logic. In *Proceedings of the 9th Belgian-Dutch Conference on Machine Learning*, Department of Computer Science, K. U. Leuven.

Rangoni, Y., & Belaïd, A. (2006). Document logical structure analysis based on perceptive cycles. In H. Bunke, & A. L. Spitz (Eds.), *Proceedings of the 7th International Workshop on Document Analysis Systems* (pp. 117-128). Berlin, Germany: Springer.

Rigamonti, M., Bloechle, J. L., Hadjar, K., Lalanne, D., & Ingold, R. (2005). Towards a canonical and structured representation of PDF documents through reverse engineering. In *Proceedings of the 8th International Conference on Document Analysis and Recognition* (pp. 1050-1055). Washington, DC: IEEE Computer Society.

Rouveirol, C. (1992). Extensions of inversion of resolution applied to theory completion. In *Inductive logic programming* (pp. 64-90). Amsterdam: Academic Press.

Sankar, K. P., Ambati, V., Pratha, L., & Jawahar, C. V. (2006). Digitizing a million books: Challenges for document analysis. In H. Bunke & A. L. Spitz (Eds.), *Proceedings of the 7th International Workshop on Document Analysis Systems* (pp. 425-436). Berlin, Germany: Springer.

Sebag, M. (1997). Distance induction in first order logic. In N. Lavrač & S. Džeroski (Eds.), In *Proceedings of the ILP-97*. Berlin, Germany: Springer.

Seki, M., Fujio, M., Nagasaki, T., Shinjo, H., & Marukawa, K. (2007). Information management system using structure analysis of paper/electronic documents and its applications. In *Proceedings of the 9th International Conference on Document Analysis and Recognition* (pp. 689-693). Washington, DC: IEEE Computer Society.

Semeraro, G., Esposito, F., Malerba, D., Fanizzi, N., & Ferilli, S. (1998). A logic framework for the incremental inductive synthesis of datalog theories. In N. E. Fuchs (Ed.), *Logic program synthesis and transformation* (LNCS 1463, pp. 300-321). Berlin, Germany: Springer.

Shi, Z., & Govindaraju, V. (2005). Multi-scale techniques for document page segmentation. In *Proceedings of the 8th International Conference on Document Analysis and Recognition* (pp. 1020-1024). Washington, DC: IEEE Computer Society.

Thompson, K., & Langley, P. (1989). Incremental concept formation with composite objects. In *Proceedings of the 6th International Workshop on Machine Learning*. San Francisco: Morgan Kaufmann.

Tuganbaev, D., Pakhchanian, A., & Deryagin, D. (2005). Universal data capture technology from semi-structured forms. In *Proceedings of the 8th International Conference on Document Analysis and Recognition* (pp. 458-462). Washington, DC: IEEE Computer Society.

Tversky, A. (1977). Features of similarity. *Psychological Review*, *84*(4), 327–352. doi:10.1037/0033-295X.84.4.327

van Beusekom, J., Keysers, D., Shafait, F., & Breuel, T. M. (2006). Distance measures for layout-based document image retrieval. In *Proceedings of the 2nd International Workshop on Document Image Analysis for Libraries* (pp. 232-242). Washington, DC: IEEE Computer Society.

van Beusekom, J., Keysers, D., Shafait, F., & Breuel, T. M. (2007). Example-based logical labeling of document title page images. In *Proceedings of the 9th International Conference on Document Analysis and Recognition* (pp. 919-923). Washington, DC: IEEE Computer Society.

Wong, K. Y., Casey, R. G., & Wahl, F. M. (1982). Document analysis system. *IBM Journal of Research and Development*, *26*(6), 647–656.

Zucker, J.-D. (1998). Semantic abstraction for concept representation and learning. In R. S. Michalski & L. Saitta (Eds.), *Proceedings of the 4th International Workshop on Multistrategy Learning* (pp. 157-164).

KEY TERMS AND THEIR DEFINITIONS

Artificial Intelligence: "the science and engineering of making intelligent machines" (J. McCarthy).

Inductive Logic Programming: a subfield of Machine Learning which uses Logic Programming as a uniform representation for examples, background knowledge and hypotheses.

Incremental Learning: a learning algorithm is incremental if it can process training examples that become available over time, usually one at a time, modifying the learned theory accordingly if necessary without restarting from scratch.

Clustering: for the aims of this work, we deliberately focus on one definition of clustering, that is the partitioning of a data set into subsets (clusters), so that the data in each subset (ideally) share some common trait – often proximity according to some defined distance measure.

Document Processing: in the scope of this work, the conversion of typed and handwritten text and images on paper-based & electronic documents into electronic information that can be easily searched and inter-related.

Document Image Understanding: "the formal representation of the abstract relationships indicated by the two-dimensional arrangement of the symbols" (G. Nagy).

Digital Libraries: a library in which collections are stored in digital formats and accessible by computers