Automatic Document Organization Exploiting FOL Similarity-based Techniques

S. Ferilli, T.M.A. Basile, M. Biba, F. Esposito
Dipartimento di Informatica - Università di Bari
via E. Orabona, 4 - 70125 Bari - Italia
{ferilli, basile, biba, esposito}@di.uniba.it

Abstract

The organization of a document collection into meaningful groups is a fundamental issue in document management systems. The grouping can be carried out by performing a comparison among the layout structure of the documents. To this aim, a powerful representation language able to describe the relations among all the document components is necessary. First-Order Logic formulæ are a powerful representation formalism characterized by the use of relations, that, however, cause serious computational problems due to the phenomenon of indeterminacy. Furthermore, a mechanism to perform the comparison among the resulting descriptions must be provided. This paper proposes the exploitation of a novel similarity formula and evaluation criteria for automatically grouping documents in a collection according to their layout structure. This is done by identifying the description components that are more similar and hence more likely to correspond to each other, based only on their syntactic structure. Experiments on a real-world dataset prove the effectiveness of the proposal.

1 Introduction

The shift from paper to digital support for documents that has happened in recent years caused a number of new opportunities, but at the same time of problems and issues, for their management and processing activities. On one hand, the digital format solved the problems of duplication and sharing that seriously affected legacy (paper) documents. On the other hand, producing new documents is nowadays so easy and cheap that a dramatic increase in the number of available documents has happened, causing the so-called *information overload* problem. In short, while the problem was formerly phisically accessing interesting documents, now the problem is accessing interesting information in the large amount of available documents. Hence, a central issue to be faced in large and heterogeneous Digital Libraries is how to organize the documents they contain into meaningful groups. This is a key point for enabling and making easier, efficient and effective document retrieval and browsing based on an overview of the documents in the collection and of their relationships.

One dimension according to which documents in a repository can be grouped and organized is according to their content. On the other hand, an alternative and, in some way, complementary approach is represented by the exploitation of the document layout similarity, since documents of the same type tend to share a similar spatial organization of their components. Indeed, the collection can be naturally organized in clusters according to the layout structure that characterizes the classes of documents, e.g. one could have newspaper articles, scientific papers and so on placed in different directories. The two approaches can be actually seen as complementary, since different kinds of content are often carried by documents having different layout styles, and that significant content is often placed in particular layout components. Thus, being able to group documents according to layout structure similarity would improve the content-based grouping as well, by pointing out the typical components of each group and suggesting to selectively read only those components for identifying their content. Since most of the existing work in the literature has been carried out in the former direction, this paper focuses on the latter one.

Manual processing of the documents in the collection is clearly unfeasible, not only for economic reasons, but also in principle, both because of the amount of documents itself and, more seriously, due to the continuing extension of the document base which is typical in real-world Digital Libraries. This motivates research for efficient and effective automatic techniques for document grouping. The wide variety of existing layout styles 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. Thus, document layout classes are hardly captured by static models, and classical attribute-value formalisms are not suitable for this task. First-order logic can over-

come these difficulties, thanks to its ability to express relationships among objects, in addition to their properties. In addition to other conceptual modelling languages, such as Entity-Relationships, it is more machineable, since learning theories as Prolog clauses allows to exploit the power of a general-purpose programming language to work on them.

In the following, after presenting fundamental concepts and related work on clustering and distance measures, a similarity framework for first-order logic clauses and their components will be discussed. Finally, its implementation and exploitation in a prototypical Digital Library Management System will be proposed, along with experiments showing how the proposed formula and criteria are able to effectively guide a clustering procedure for FOL descriptions in a real-world document collection.

2 Preliminaries and Related Work

The *Clustering* task aims at organizing a collection of unlabeled 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. Many research efforts on data representation, elements' similarity and grouping strategies have produced several successful clustering methods (see [14] for a survey). Closely related to data clustering is Conceptual Clustering, a Machine Learning paradigm for unsupervised classification which aims at generating a concept description for each generated class. In conceptual clustering both the inherent structure of the data and the description language, available to the learner, drive cluster formation. Thus, a concept (regularity) in the data could not be learned by the system if the description language is not powerful enough to describe that particular concept (regularity). This problem arises when the elements simultaneously describe several objects whose relational structures change from one element to the other. First-Order Logic representations allow to overcome these problems.

First-order logic (*FOL*) is a powerful formalism able to express relations between objects and hence can overcome the typical limitations shown by propositional or attribute-value representations. As a consequence and tradeoff for their expressive power, the presence of relations causes various portions of one description to be possibly mapped in (often many) different ways onto another description (which is known as *indeterminacy*). This poses serious problems of computational effort when two descriptions have to be compared to each other in order check whether the former entails the latter or to assess their similarity. Hence, the need for a set of general criteria that are able to support the comparison and guide the mapping between formulæ. Specifically, our objective is developing a similarity framework for FOL descriptions, based on a measure and a number of criteria that must be simple (in order to ensure efficient computation), sensible (they have to reflect as much as possible the intuitive way in which humans compare two descriptions), effective (must capture as much information as possible about the items to be compared), flexible (allow to weight differently the two items) and general (are not devised *ad-hoc* for some problem).

An interesting perspective on FOL is that, by restricting to formulæ in the form of clauses, it defines Logic Programming [18], a paradigm in which program execution is cast as finding a proof for some query expressed as a conjunction of atomic formulae, and hence provides the key for making logic a machineable formalism (note that any FOL formula can be expressed as a set of clauses, after rewriting it in Conjunctive Normal Form). Logic programs are made up of *Horn clauses*, i.e. logical formulæ of the form $l_1 \wedge \cdots \wedge l_n \Rightarrow l_0$ where the l_i's are *atoms* (each of which is a predicate, applied to a number of terms equal to its arity), to be interpreted as "l_0 (*head*) is true, provided that l_1 and ... and l_n (*body*) are all true". Specifically, it is possible to focus without loss of generality to linked Datalog clauses (a restriction in which only variables and constants are allowed as terms): indeed, linked sub-parts of non-linked clauses can be dealt with separately, having no connection between each other, while the *flattening/unflattening* procedures [24] can translate generic first-order clauses into Datalog ones and vice-versa.

Despite of a large number of similarity/distance measures and techniques developed for attribute-value representations [16], few works are present in the literature for first-order descriptions. [10] proposes a distance measure between structural symbolic descriptions based on probability theory applied to the formula components; in addition to intuition and computation issues, it requires simplifying hypotheses (statistical independence, mutual exclusion) to ease the probability handling and *a-priori* knowledge of the representation language, it is based on the presence of 'mandatory' relations and it is limited to a Model-Observation comparison.

Many supervised learning systems are based on a distance measure. In *KGB* [2] the similarity computation is not straightforward and negative information cannot be handled. *RIBL* [7] is a k-Nearest Neighbor classifier based on a modified version of the FOL similarity function proposed in [2] in which the similarity between objects depends on the similarity of their attributes' values and of the objects related to them and so on, which poses the problem of indeterminacy in associations. In *RISE* [6], when an instance to be classified is not covered by any rule, it is classified according to the majority class among its "nearest" (i.e., most similar) rules. [25] presents an approach in which, given an example and k clauses, the example coverage by

each clause is used as a distinct dimension in a k-ary space on which computing a distance. The approach in [20] proposes a distance between terms based on a level mapping (a function that maps every simple expression on a natural number) where the main functor is most important and the deeper nested subterms are less important. [22] presents a distance function between two atoms, based on the difference with their least general generalization, that consists of a pair whose first component is an extension of the notion of distance used in [20], and the second component allows to differentiate distances that have the same value for the first component. The approach proposed by Kodratoff [15] is based on just an evaluation of similarity between objects, whose overall most likely associations are then set so to maximize the global fitness.

However, most clustering algorithms and systems work on attribute-value representation (e.g., `CLUSTER/2` [19], `CLASSIT` [13], `COBWEB` [12]). Other systems such as `LABYRINTH` [26] can deal with structured objects exploiting a representation that is not powerful enough to express the dataset in many domains. There are few systems that cluster examples represented in FOL (e.g., `AUTOCLASS-like` [23], `KBG` [1]), some of which still rely on propositional distance measures (e.g., `TIC` [3]).

3 Similarity Parameters and Formula

In the particular case of document processing, attribute-value representations and techniques are not always suitable because documents in the same layout class 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 even vary significantly. This could be the case, for instance, of the title block, whose length could range from 1 to 4 lines. Thus, handling relations can be a key advantage to capture the page layout invariants that characterize document classes, and additionally produces human-readable descriptions that can be exploited by domain experts for validation or understanding purposes.

Once FOL has been chosen as a sufficiently powerful representation language to describe the layout structure of documents, corresponding processing tools are needed to handle the document descriptions in order to learn from them. While standard FOL reasoning mechanisms are SLD-resolution and θ-subsumption, they only allow for binary answers on whether a given (more general) formula accounts for another (more specific) one. This section summarizes a mechanism to assess the similarity among first-order (specifically, Horn-clause) logic descriptions, developed to overcome the problem and applicable to any domain, not just document processing. This means that it must be general enough, and cannot be based on particular interpretations of predicates, but must work on the syntactic aspects alone. Indeed, it has been exploited in other application domains (such as Mutagenesis, in which descriptions concern chemical compounds whose mutagenicity must be predicted according to their chemical structure) and on other tasks (such as supervised inductive generalization [11]) as well.

The evaluation of similarity between two items i' and i'' might be based both on the presence of common features, which should concur in a positive way to the similarity evaluation, and on the features of each item that are not owned by the other, which should concur negatively to the whole similarity value assigned to them [17]. Thus, plausible parameters on which basing similarity between i' and i'' are:

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').

We developed a novel similarity function, $sf(\alpha, i', i'')$, that intuitively expresses the degree of similarity between two items i' and i'' based on the above parameters, also including an additional parameter α, $0 \leq \alpha \leq 1$, that weights the importance of either item. Thus, $\mathrm{sf}(\alpha, i', i'')$ can be written as $\mathrm{sf}(\alpha, n, l, m)$, where n, l and m are the similarity parameters referred to i' and i''. In the following we will use both forms indifferently, as needed.

$$\mathrm{sf}(\alpha, i', i'') = \mathrm{sf}(\alpha, n, l, m) =$$

$$= \alpha \frac{l+1}{l+n+2} + (1-\alpha)\frac{l+1}{l+m+2} \qquad (1)$$

In FOL formulæ, 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 between the objects referred to in the descriptions, and the *structure* one, referring to how the nets of relationships in the descriptions overlap.

Given two clauses C' and C''. Call $A' = \{a'_1, \ldots, a'_n\}$ and $A'' = \{a''_1, \ldots, a''_m\}$ the set of terms in C' and C''. We may think of comparing a pair $(a', a'') \in A' \times A''$, made up of an object taken from C' and one taken from C'', respectively. Intuitively, for evaluating the similarity between two objects, one would like the overlapping between their features to be as large and complete as possible, and their residual to be the smallest. Informally, two kinds of object features can be distinguished: the properties they own (let us call these *characteristic features*), usually expressed in first-order logic by unary predicates (e.g. `type_text(X)`), and

the ways in which they relate to other objects (which we call *relational features*), generally expressed by n-ary predicates (e.g. `on_top(X,Y)`). More precisely, relational features are defined by the position the object holds among the n-ary predicate arguments, since different positions actually refer to different roles played by the objects. For instance, in the predicate `on_top(X,Y)` the first argument position identifies the role of the upper object and the second one represents the role of the lower one.

Two corresponding similarity values can be associated to a' and a'': a *characteristic similarity*, where (1) is applied to values related to the characteristic features, and a *relational similarity*, based on how many times the two objects play the same or different roles in the n-ary predicates.

To take into account the similarity between a' and a'' based on their characteristic features (*characteristic similarity*), let us consider the set P' of properties related to a' and the set P'' of properties related to a''. Now, the characteristic similarity between a' and a'' can be computed as

$$\mathrm{sf}_c(a', a'') = \mathrm{sf}(n_c, l_c, m_c)$$

for the following parameters:

$n_c = |P' \setminus P''|$ is the number of properties owned by a' in C' but not by a'' in C'' ;

$l_c = |P' \cap P''|$ is the number of common properties between a' in C' and a'' in C'';

$m_c = |P'' \setminus P'|$ is the number of properties owned by a'' in C'' but not by a' in C' .

A similar technique can be applied to compute the degree of similarity between a' and a'' according to their relational features (*relational similarity*). In this case, being interested in the roles played by an object, due to the possibility that the same object plays multiple times the same role in different relations (e.g., an object laying on many others), we have to take into account the number of occurrences, and hence consider the *multi*set R' of roles played by a' and the *multi*set R'' of roles played by a''. Then, the relational similarity between a' and a'' can be computed as

$$\mathrm{sf}_r(a', a'') = \mathrm{sf}(n_r, l_r, m_r)$$

for the following parameters:

$n_r = |R' \setminus R''|$ expresses how many times a' plays in C' role(s) that a'' does not play in C'' ;

$l_r = |R' \cap R''|$ is the number of times that both a' in C' and a'' in C'' play the same role(s);

$m_r = |R'' \setminus R'|$ expresses how many times a'' plays in C'' role(s) that a' does not play in C'.

Overall, we can define the *object similarity* between two terms as

$$\mathrm{sf}_o(a', a'') = \mathrm{sf}_c(a', a'') + \mathrm{sf}_r(a', a'') \quad (2)$$

that ranges in $]0, 2[$ (but can obviously be normalized to $]0, 1[$ if needed).

When checking for the structural similarity of two formulæ, many objects can be involved, and hence their mutual relationships represent a constraint on how each of them in the former formula can be mapped onto another in the latter. Differently from the case of objects, what defines the structure of a formula is the set of n-ary predicates, and specifically the way in which they are applied to the various objects to relate them (i.e., *atoms*, according to the terminology introduced above). This is the most difficult part, since relations are specific to the first-order setting and are the cause of indeterminacy in mapping (parts of) a formula into (parts of) another one. In the following, we will call *compatible* two FOL (sub-)formulæ 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* $\mathrm{sf}_s(l', l'')$ between two compatible n-ary atoms l' and l'' having stars S' and S'', respectively, can be computed based on the number of common and different elements in each of the two stars, represented by the following parameters:

$n_s = |S' \setminus S''|$ expresses how many more relations l' has in C' than l'' has in C'' ;

$l_s = |S' \cap S''|$ is the number of relations that both l' in C' and l'' in C'' have in common;

$m_s = |S'' \setminus S'|$ expresses how many more relations l'' has in C'' than l' has in C'.

Since this value refers only to atom-related information, and does not take into account the similarity of the involved objects, an overall more adequate evaluation of similarity between l' and l'' can be obtained by taking into account also the object similarity values for all pairs of terms included in the association θ that map l' onto l'' of their arguments in corresponding positions:

$$\mathrm{sf}_s(l', l'') = \mathrm{sf}(n_s, l_s, m_s) + C^s(\{\mathrm{sf}_o(t', t'')\}_{t'/t'' \in \theta}) \quad (3)$$

where C^s is a composition function (e.g., the average). In case sf_o ranges in $]0, 2[$ it ranges in $]0, 3[$, but can obviously be normalized to $]0, 1[$ if needed.

Given a clause C, we define its *associated graph* as $G_C = (V, E)$ with

$V = \{l_0\} \cup \{l_i | i \in \{1, \ldots, n\}, l_i$ built on k-ary predicate, $k > 1\}$ and

$E \subseteq \{(a_1, a_2) \in V \times V \mid terms(a_1) \cap terms(a_2) \neq \emptyset\}$

where $terms(a)$ denotes the set of terms that appear as arguments of atom a. The strategy for choosing the edges to be represented is based on the idea that we can leverage on the presence of a single atom in the head to have both a starting point and precise directions for traversing the graph in order to choose a unique and well-defined perspective on the clause structure among the many possible: indeed, the heads somehow represent the intended meaning of the body (e.g., the object to be classified or clustered), and must necessarily match to each other (being the only head atom in the clause). More precisely, we build a Directed Acyclic Graph, *stratified* (i.e., with the set of nodes partitioned) in such a way that, for any fixed stratum (element of the partition), all the incoming edges come from a single (different) stratum, and all the outcoming edges reach a single stratum (different from both the previous ones), as follows. The head is the only node at level 0 (first element of the partition). Then, each successive level (element of the partition) is made up by 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.

Now, all possible paths starting from the head and reaching *leaf* nodes (those with no outcoming edges) can be interpreted as the basic components of the overall structure of the clause, and be exploited instead of single atoms when checking similarity between clauses. Being such paths univoquely 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.

Definition Given two clauses C' and C'' with associated graphs $G_{C'}$ and $G_{C''}$ respectively, and two paths
$p' =< l'_0, l'_1, \ldots, l'_{n'} >$ in $G_{C'}$ and
$p'' =< l''_0, l''_1, \ldots, l''_{n''} >$ in $G_{C''}$.
The *intersection* between p' and p'' is defined as the pair of longest compatible initial subsequences of p' and p'', excluding the head:

$p' \cap p'' = (p_1, p_2) = (< l'_1, \ldots, l'_k >, < l''_1, \ldots, l''_k >)$
s.t. $\forall i = 1, \ldots, k$:
$l'_1, \ldots, l'_i$ compatible with $l''_1, \ldots, l''_i \wedge$
$(k = n' \vee k = n'' \vee l'_1, \ldots, l'_{k+1}$ incompatible with $l''_1, \ldots, l''_{k+1})$

The *differences* between p' and p'' are then defined as the incompatible trailing parts:

$p' \setminus p'' =< l'_{k+1}, \ldots, l'_{n'} >$
$p'' \setminus p' =< l''_{k+1}, \ldots, l''_{n''} >$ $\diamond$

Hence, the *path similarity* between p' and p'', $\mathrm{sf}_s(p', p'')$, is computed by applying (1) to the following parameters:

$n_p = |p' \setminus p''| = n' - k$ is the length of the trail incompatible sequence of p' wrt p'' ;

$l_p = |p_1| = |p_2| = k$ is the length of the maximum compatible initial sequence of p' and p'';

$m_p = |p'' \setminus p'| = n'' - k$ is the 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:

$$\mathrm{sf}_p(p', p'') = \mathrm{sf}(n_p, l_p, m_p) + C^p(\{\mathrm{sf}_s(l'_i, l''_i)\}_{i=1,\ldots,k}) \quad (4)$$

where C^p is a composition function (e.g., the average). In case sf_s ranges in $]0, 3[$ it ranges in $]0, 4[$, but can obviously be normalized to $]0, 1[$ if needed.

4 Clause Similarity

The overall similarity between two (tuples of) terms reported in the head predicates of two clauses, according to their description reported in the respective bodies, can be computed based on their generalization. In particular, one would like to exploit their *least general generalization* (lgg), i.e. the most specific model for the given pair of descriptions. Unfortunately, such a generalization is not easy to find: either classical θ-subsumption is used as a generalization model, and then one can compute Plotkin's lgg [21], at the expenses of some undesirable side-effects concerning the need of computing its reduced equivalent, or one requires the generalization to be a subset of the clauses to be generalized. In the latter option, that we choose for the rest of the work, the θ_{OI} generalization model [8] represents a supporting framework with solid theoretical foundations to be exploited.

Given two clauses C' and C'', call $C = \{l_1, \ldots, l_k\}$ their lgg, and consider the substitutions θ' and θ'' such that $\forall i = 1, \ldots, k : l_i\theta' = l'_i \in C'$ and $l_i\theta'' = l''_i \in C''$, respectively. Thus, a formula for assessing the overall similarity between C' and C'', called *formulæ similitudo* and denoted *fs*, can be computed according to the amounts of common and different atoms:

$n = |C'| - |C|$ 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 of common and different objects:

$n_o = |terms(C')| - |terms(C)|$ how many terms in C' are not associated by its least general generalization to terms in C'';

$l_o = |terms(C)|$ maximal number of terms that can be put in correspondence in C' and C'' as associated by their least general generalization;

$m_o = |terms(C'')| - |terms(C))|$ how many terms in C'' are not associated by its least general generalization to terms in C'.

by considering also the star similarity values for all pairs of atoms associated by the least general generalization:

$$\mathrm{fs}(C', C'') = \mathrm{sf}(n, l, m) \cdot \mathrm{sf}(n_o, l_o, m_o) + \\ + C^c(\{\mathrm{sf}_s(l'_i, l''_i)\}_{i=1,\ldots,k})$$

where C^c is a composition function (e.g., the average). In case sf_s ranges in $]0, 2[$ it ranges in $]0, 3[$, but 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 correspondence (which includes both structural and object similarity), smoothed by adding the overall similarity in the number of overlapping and different atoms and objects between the two (whose weight in the final evaluation should not overwhelm the similarity coming from the detailed comparisons, hence the multiplication).

The proposed approach overcomes some problems that are present in related work. Compared to [10], our function does not require the assumptions and simplifying hypotheses (statistical independence, mutual exclusion) to ease the probability handling, and no *a-priori* knowledge of the representation language is required (such as the type domains). It does not require the user to set weights on the predicates' importance, and is not based on the presence of 'mandatory' relations, like for the $G1$ subclause in [10]. With respect to [2], it can be easily extended to handle negative information. With respect to [7], it avoids the propagation of similarity between sub-components that poses the problem of indeterminacy in associations. Compared to [22] it yields a unique value as a result of a comparison, which is more understandable and comfortable for handling. With respect to [25] it is based directly on the structure, and not on derived features.

5 Exploitation

In document processing systems, three typical phases are carried out to identify the significant components of a digital document. *Layout Analysis* consists in the perceptual organization process that aims at identifying the single blocks of a document and at detecting relations among them (*Layout Structure*); then, associating the proper logical role to each component yields the *Document Logical Structure*. Since the logical structure is different according to the kind of document, this is obtained in two steps: *Document Image Classification*, aiming at categorizing the document (e.g., newspaper, scientific paper, email, technical report, call for papers), and *Document Image Understanding*, aiming identificating of the significant layout components for that class and at associating each of them to a tag that expresses its role (e.g., signature, object, title, author, abstract, footnote, etc.). We propose to apply multistrategy Machine Learning techniques along these phases of document processing where the classical statistical and numerical approaches to classification and learning may fail, being not able to deal with the lack of a strict layout regularity in the variety of documents available online.

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

The continuous flow of new and different documents in a Web repository or in Digital Libraries calls for *incremental* abilities of the system, that must be able to update or revise a faulty knowledge previously acquired for identifying the logical structure of a document. Traditionally, Machine Learning methods that automatically acquire knowledge in developing intelligent systems, require to be provided with a set of training examples, belonging to a defined number of classes, and exploit them altogether in a batch way. Thus, classical approaches require that the number of classes is defined and fixed since the beginning of the induction step: this prevents the opportunity of dealing with totally new instances, belonging to new classes, that require the ability to incrementally revise a domain theory as soon as new data are encountered. Indeed, Digital Libraries require autonomous or semi-autonomous operation and adaptation to

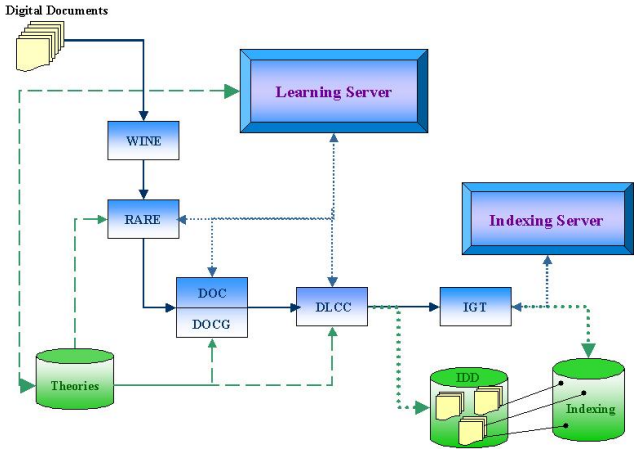

Figure 1. Document Management System Architecture

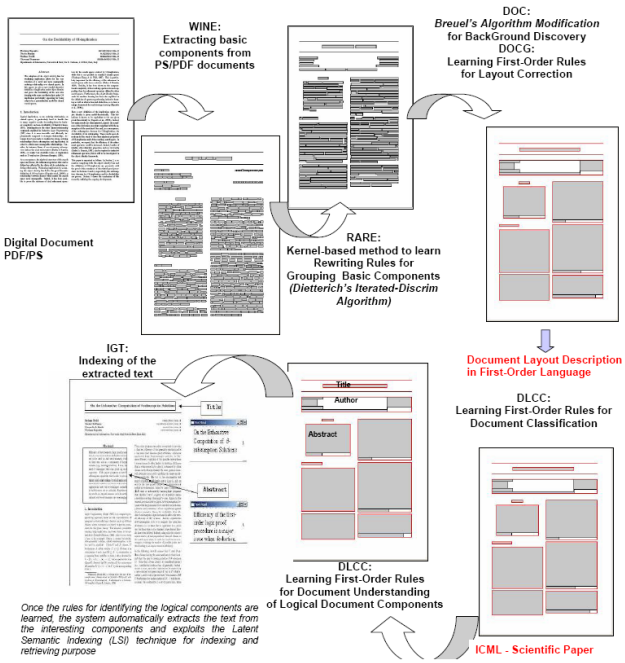

Figure 2. Document Management System Process Flow

changes in the domain, the context, or the user needs. If any of these changes happens, the classical approach requires that the entire learning process is restarted to produce a model capable of coping with the new scenario. Such requirements suggest that incremental learning, as opposed to classical batch one, is needed whenever either incomplete information is available at the time of initial theory generation, or the nature (and the kinds) of the concepts evolves dynamically. E.g., this is the case of modifications in time of typing style of documents that nevertheless belong to the same class or of the introduction of a completely new class. Incremental processing allows for continuous responsiveness to the changes in the context, can potentially improve efficiency and deals with concept evolution. The incremental setting implicitly assumes that the information (observations) gained at any given moment is incomplete, and thus that any learned theory could be susceptible of changes.

DOMINUS (DOcument Management INtelligent Universal System) is characterized by the intensive exploitation of intelligent techniques in each step of document processing from acquisition to indexing, from categorization to storing and retrieval. Since it is general and flexible, it can be embedded as a document management engine into many different Digital Library systems. The overall architecture of DOMINUS is reported in Figure 1. A central role is played by the Learning Server, which intervenes during different processing steps in order to continuously adapt the acquired knowledge taking into consideration new experimental evidence and changes in the context. The corresponding process flow performed by the system from the original digital documents acquisition to text extraction and indexing is reported in Figure 2.

The layout analysis process on documents in digital format starts with the application of a pre-processing module, called WINE (Wrapper for the Interpretation of Non-uniform Electronic document formats), that takes as input a digital document and produces (by an intermediate vector format) the initial document's XML basic representation, that describes it as a set of pages made up of basic blocks. Due to the large number of basic blocks discovered by WINE, that often correspond to fragments of words (e.g., in PostScript and Portable Document Format documents), it is necessary a first aggregation based on blocks overlapping or adjacency, yielding composite blocks corresponding to whole words. The number of blocks after this step is still large, thus a further aggregation (e.g., of words into lines) is needed. Since grouping techniques based on the mean distance between blocks proved unable to correctly handle the case of multi-column documents, such a task was cast to a multiple instance problem and solved exploiting the kernel-based method proposed in [5], implemented in the Learning Server module, that is able to generate rewriting rules that suggest how to set some parameters in order to group together word blocks to obtain lines. The inferred rules will be stored in the Theories knowledge base for future exploitation by RARE (Rule Aggregation REwriter) and modification. A machine learning approach is needed in this task because basing word grouping on a fixed inter-word spacing size would not ensure correct handling of multi-column documents (in some cases two words belonging to co-linear lines in adjacent columns could be

merged into a single line).

Once such a line-block representation is generated, DOC (Document Organization Composer) collects the semantically related blocks into groups by identifying the surrounding frames. This step is based on a background structure analysis that first retrieves white spaces according to Breuel's algorithm [4], that finds iteratively the maximal white rectangles in a page, and then computes the content block as the complement of the background. Actually, Breuel's original algorithm was modified in order to make it more efficient and to avoid retrieving insignificant background white spaces such as inter-word or inter-line ones, by forcing the process to stop before its natural conclusion. At the end of this step, some blocks might not be correctly recognized, because of significant white spaces not recognized by the background analysis or, conversely, because of insignificant ones being considered. In such cases a phase of layout correction is needed, that is automatically performed in DOCG (Document Organization Correction Generator) by exploiting embedded rules stored in the Theories knowledge base. Such rules were automatically learned by the Learning Server from previous manual corrections performed by the users on the first documents and collected by the system. Since the significance of a background rectangle is determined by its relative size and position with respect to the other content and background rectangles around it, a first-order logic (relational) learning approach is exploited to infer rules for the elimination of background rectangles erroneously retrieved by the system, or for the addition of those erroneously ignored.

Once the layout structure has been correctly and definitely identified, the document class must be recognized and a semantic role must be associated to each significant component (*Document Image Understanding*) in order to perform the automatic extraction of the interesting text with the aim of improving document indexing. At the end of this step both the original document and its XML representation, enriched with class information and components annotation, is stored in the Internal Document Database, IDD. Finally, the text is extracted from the significant components and the Indexing Server is called by the IGT (IndexinG of Text) module to manage such information, useful for an effective content-based document retrieval.

The Document Image Understanding task, on which this work is particularly focused, is performed by DLCC (Document and Layout Components Classifier) by firstly associating the document to a class that expresses its type and then associating to every significant layout component a tag expressing its role. Both these steps are performed thanks to first-order logic theories previously learned and stored in the Theories knowledge base. In case of failure these theories can be properly updated without restarting learning from scratch, this way improving efficiency in obtaining a new operational theory. Indeed, the theory revision step is performed by a first-order incremental learning system that runs on the new observations and tries to modify the old theories in the knowledge base. It is worth noting that such a system is already able to refine the theories not only by generalizing or specializing definitions for existing classes, but even by adding completely new classes whenever their instances become available. This is a fundamental feature in an evolving domain such as a digital document repository.

The novelty introduced in this paper is the extension of the DOMINUS' Learning Server module with a clustering functionality that can, when needed, group documents that could not be associated to any known class into homogeneous groups, each of which could become a new class itself (for which specific classification rules can be learned), this way dynamically extending the set of known classes. More specifically, the idea consists in collecting the documents rejected by the current theory and for which no class information is provided, and using them for inferring the new classes when a sufficient amount is available.

6 Experimental evaluation

Some experiments were designed to check whether the proposed framework is actually able to provide significant groups of documents in an unsupervised setting based on their layout descriptions. All of them were run under WindowsXP Professional on a PC endowed with a 2.13 GHz Intel processor and 2GB RAM. To evaluate such a performance, a dataset for which the document classes were already known was used, by hiding each document's class to the system and checking whether it was able to autonomously infer the correct groups.

The dataset concerns layout descriptions of first pages of scientific papers [9], automatically generated by DOMINUS, according to which identifying the papers' series. In detail, it contains 353 descriptions, belonging to 4 different classes: Elsevier journals, Springer-Verlag Lecture Notes (SVLN) series, Journal of Machine Learning Research (JMLR) and Machine Learning Journal (MLJ). The complexity of the dataset is considerable, due to several aspects: the journals layout styles are quite similar, so that it is not easy to grasp the difference when trying to group them in distinct classes; moreover, the 353 documents are described with a total of 67920 atoms, for an average of more than 192 atoms per description (some descriptions are made up of more than 400 atoms); last, the description is heavily based on a membership relation that increases indeterminacy and hence can stress significantly the similarity computation.

The aim of the experiment was to generate clusters of the given document collection. The clustering task aims at

grouping a set of items into homogeneous classes according to the similarity between their descriptions. In a next step, definitions for each cluster can be inferred and exploited to classify future documents (*conceptual* clustering). The clustering algorithm exploited for the experiment consisted in a classical K-means algorithm, exploiting *medoid* prototypes 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 formulæ do not induce an euclidean space. The stop criterion was set as the moment in which a new iteration outputs a partition already seen in previous iterations.

As already noted, since the class of each observation in the datasets is known, for correct evaluation purposes the correct number of clusters to be obtained was provided to the clustering procedure, and then the resulting clusters were compared with the correct classes to assess their quality (*supervised clustering*). Specifically, each cluster was compared to its best-matching class, and their overlapping evaluated according to precision, recall and purity. In fact, for each cluster the precision-recall values were neatly high for one class and considerably low for all the others; moreover, each cluster had a different best-matching class, so that the association and consequent evaluation became straightforward.

Results, summarized in Table 1, show a 91.60% precision (Prec), 93.28% recall (Rec) and 92.35% purity (Pur), indicating that the proposed method is highly effective since it is able to autonomously recognize the original classes. This is very encouraging, considering that precision and recall are typically contrasting parameters, and especially in the perspective of the representation-related difficulties. The clusters cardinality resembles very much that of the corresponding actual class. More in-depth analysis reveals that clusters associated to Elsevier and JMLR classes include all of the correct documents, but only in the case of Elsevier at the cost of a lower precision. On the other hand, the clusters corresponding to SVLN and MLJ show a very similar behaviour, with a high precision balanced by a slightly lower recall. I any case, it should be taken into account that some of the classes in the dataset are very similar among each other as regards the layout style. As to the time performance, it was in the order of hours, but this is not a hard problem since document clustering is done off-line and the new classes are made available only after the procedure has been accomplished.

Table 1. Experimental results of the proposed approach on Document Clustering

Class	|Cluster|	|Class|	**Prec (%)**	**Rec (%)**
Elsevier	65	52	80	100
SVLN	65	75	98,46	85,33
JMLR	105	95	90,48	100
MLJ	118	131	97,46	87,79
Pur = 92,35%	Total = *353*		**91,60**	**93,28**

7 Conclusions

The problem of organizing a document collection into meaningful groups is a fundamental issue in Digital Libraries for providing efficient and effective funcionalities such as browsing and indexing. In order to automatically perform such a grouping, it is possible to leverage on the layout structure to cluster the documents into homogeneous classes according to the similarity between their layout. To this aim, First-Order Logic formulæ is proposed as a sufficiently powerful representation formalism, able to describe the relations among document components. However, this requires a different approaches to similarity than classical attribute-value based representations. This paper based clustering on a novel similarity framework in order to automatically group documents of unknown class in a collection, this way extending the set of known classes. In order to embed such a procedure in the DOMINUS document management system, an experimental validation of the approach has been performed. Results show high precision and recall values in autonomously identifying the groups of known documents without being provided with their classes. Hence, a motivation for continuing in this direction.

Future work will concern experimentation on different document types, an extension of the approach to autonomously infer the correct number of clusters in which distributing the available documents and a tighter integration in DOMINUS.

References

[1] G. Bisson. Conceptual clustering in a first order logic representation. In *ECAI '92: Proceedings of the 10th European conference on Artificial intelligence*, pages 458–462. John Wiley & Sons, Inc., 1992.

[2] G. Bisson. Learning in FOL with a similarity measure. In W. Swartout, editor, *Proc. of AAAI-92*, pages 82–87, 1992.

[3] H. Blockeel, L. D. Raedt, and J. Ramon. Top-down induction of clustering trees. In J. Shavlik, editor, *Proceedings of the 15th International Conference on Machine Learning*, pages 55–63. Morgan Kaufmann, 1998.

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

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

[6] P. Domingos. Rule induction and instance-based learning: a unified approach. In *Proc. of IJCAI-95*, pages 1226–1232. Morgan Kaufmann, 1995.

[7] W. Emde and D. Wettschereck. Relational instance based learning. In L. Saitta, editor, *Proc. of ICML-96*, pages 122–130, 1996.

[8] F. Esposito, N. Fanizzi, S. Ferilli, and G. Semeraro. A generalization model based on oi-implication for ideal theory refinement. *Fundam. Inform.*, 47(1-2):15–33, 2001.

[9] F. Esposito, S. Ferilli, T. Basile, and N. D. Mauro. Machine learning for digital document processing: from layout analysis to metadata extraction. In S. Marinai and H. Fujisawa, editors, *Machine Learning in Document Analysis and Recognition*, volume 90 of *Studies in Computational Intelligence*, pages 105–138. Springer Verlag - ISBN: 978-3-540-76279-9, 2008.

[10] F. Esposito, D. Malerba, and G. Semeraro. Classification in noisy environments using a distance measure between structural symbolic descriptions. *IEEE Transactions on PAMI*, 14(3):390–402, 1992.

[11] S. Ferilli, T. Basile, N. D. Mauro, M. Biba, and F. Esposito. Similarity-guided clause generalization. In *AI*IA-2007: Artificial Intelligence and Human-Oriented Computing*, volume 4733 of *LNAI*, pages 278–289. Springer, 2007.

[12] D. H. Fisher. Knowledge acquisition via incremental conceptual clustering. *Machine Learning*, 2(2):139–172, 1987.

[13] J. H. Gennari, P. Langley, and D. Fisher. Models of incremental concept formation. *Artificial Intelligence*, 40(1-3):11–61, 1989.

[14] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM Computing Surveys*, 31(3):264–323, 1999.

[15] Y. Kodratoff and J.-G. Ganascia. Improving the generalization step in learning. In R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, editors, *Machine Learning: An Artificial Intelligence Approach: Volume II*, pages 215–244. Kaufmann, Los Altos, CA, 1986.

[16] M. Li, X. Chen, X. Li, B. Ma, and P. Vitanyi. The similarity metric, 2003.

[17] D. Lin. An information-theoretic definition of similarity. In *Proc. 15th International Conf. on Machine Learning*, pages 296–304. Morgan Kaufmann, San Francisco, CA, 1998.

[18] J. W. Lloyd. *Foundations of logic programming; (2nd extended ed.)*. Springer-Verlag New York, Inc., USA, 1987.

[19] R. S. Michalski and R. E. Stepp. Learning from observation: Conceptual clustering. In R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, editors, *Machine Learning: An Artificial Intelligence Approach*, pages 331–363. Springer: Berlin, 1984.

[20] S. Nienhuys-Cheng. Distances and limits on herbrand interpretations. In D. Page, editor, *Proc. of ILP-98*, volume 1446 of *LNAI*, pages 250–260. Springer, 1998.

[21] G. D. Plotkin. A note on inductive generalization. *Machine Intelligence*, 5:153–163, 1970.

[22] J. Ramon. *Clustering and instance based learning in first order logic*. PhD thesis, Dept. of Computer Science, K.U.Leuven, Belgium, 2002.

[23] J. Ramon and L. Dehaspe. Upgrading bayesian clustering to first order logic. In *Proceedings of the 9th Belgian-Dutch Conference on Machine Learning*, pages 77–84. Department of Computer Science, K.U.Leuven, 1999.

[24] C. Rouveirol. Extensions of inversion of resolution applied to theory completion. In *Inductive Logic Programming*, pages 64–90. Academic Press, 1992.

[25] M. Sebag. Distance induction in first order logic. In N. Lavrač and S. Džeroski, editors, *Proc. of ILP-97*, volume 1297 of *LNAI*, pages 264–272. Springer, 1997.

[26] K. Thompson and P. Langley. Incremental concept formation with composite objects. In *Proceedings of the sixth international workshop on Machine learning*, pages 371–374. Morgan Kaufmann Publishers Inc., 1989.