Generalization-based Similarity for Conceptual Clustering

S. Ferilli, T.M.A. Basile, N. Di Mauro, M. Biba, and F. Esposito

Dipartimento di Informatica
Università di Bari
via E. Orabona, 4 - 70125 Bari - Italia
{ferilli, basile, ndm, biba, esposito}@di.uniba.it

Abstract. The availability of techniques for comparing descriptions has many applications in Artificial Intelligence, ranging from description selection to flexible matching, from instance-based learning to clustering. Due to the complexity of handling First-Order Logic formulæ, where the presence of relations causes various portions of one description to be possibly mapped in different ways onto another description, few works are available in the literature for this kind of representations.
This paper tackles the case of Conceptual Clustering, where a new approach to similarity evaluation, based on both syntactic and semantic features, is exploited to support the task of grouping together similar items according to their relational description.
After presenting a framework for Horn Clauses (including criteria, a function and composition techniques for similarity assessment), classical clustering algorithms are exploited to carry out the grouping task. Experimental results on real-world datasets prove the effectiveness of the proposal.

1 Introduction

First-order logic (*FOL* for short) is a powerful formalism, that is able to express relations between objects and hence can overcome the limitations shown by propositional or attribute-value representations. However, the presence of relations causes various portions of one description to be possibly mapped in different ways onto another description, which poses problems of computational effort when two descriptions have to be compared to each other. Hence, the availability of techniques for the comparison between FOL (sub-)descriptions could have many applications, particularly in the Artificial Intelligence community: helping a subsumption procedure to converge quickly, guiding a generalization procedure by focussing on the components that are more similar and hence more likely to correspond to each other, implementing a *flexible matching*, supporting instance-based classification techniques or *conceptual clustering*.

Specifically, an important subclass of FOL refers to sets 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*, usually represented in Prolog style as l_0 :- $l_1, \ldots, l_n$ to be interpreted as "l_0 (called

head of the clause) is true, provided that l_1 and ... and l_n (called *body* of the clause) are all true". Without loss of generality [17], we will deal with the case of linked Datalog clauses.

In the following sections, some criteria and a formula on which basing similarity considerations between first-order logic clauses will be presented, that are intended to represent a good tradeoff between significance, effectiveness and expressiveness on one side, and computational efficiency on the other. Then, Section 5 will show how the proposed formula and criteria, are able to effectively guide a clause generalization procedure. Lastly, Section 4 will deal with related work, while 6 will conclude the paper and outline future work directions.

2 Similarity Formula

Intuitively, 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 [11]. Thus, plausible similarity parameters 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').

A novel similarity function that expresses the degree of similarity between i' and i'' based on the above parameters, developed to overcome some limitations of other functions in the literature (e.g., Tverski's, Dice's and Jaccard's), is the following:

$$sf(i', i'') = \mathrm{sf}(n, l, m) = 0.5\frac{l+1}{l+n+2} + 0.5\frac{l+1}{l+m+2} \tag{1}$$

It takes values in $]0, 1[$, which resembles the theory of probability and hence can help human interpretation of resulting value. A complete overlapping of the two items tends to the limit of 1 as long as the number of common features grows. The full-similarity value 1 is never reached, and is reserved to the exact identification of items, i.e. $i' = i''$ (in the following, we assume $i' \neq i''$). Conversely, in case of no overlapping the function will tend to 0 as long as the number of non-shared features grows. This is consistent with the intuition that there is no limit to the number of different features owned by the two descriptions, which contribute to make them ever different. It evaluates to $1/2$ in case of no features at all in two descriptions ($n = l = m = 0$), which is intuitive for a case of maximum uncertainty. Since each of the two terms refers specifically to one of the two clauses under comparison, a weight could be introduced to give different importance to either of the two.

3 Similarity Criteria

The main contribution of this paper is in the exploitation of the formula in various combinations that can assign a similarity degree to the different clause constituents. In FOL formulæ, terms represent specific objects; unary predicates represent term properties and n-ary predicates express relationships. Hence, two levels of similarity between first-order descriptions can be defined: the *object* level, concerning similarities between terms in the descriptions, and the *structure* one, referring to how the nets of relationships in the descriptions overlap.

Example 1. Let us consider, as a running example throughout the paper, the following two clauses (in this case, a rule C and a classified observation E):

$$C : h(X) \text{ :- } p(X,Y), p(X,Z), p(W,X), r(Y,U), o(Y,Z), q(W,W), s(U,V), \\ \pi(X), \phi(X), \rho(X), \pi(Y), \sigma(Y), \tau(Y), \phi(Z), \sigma(W), \tau(W), \pi(U), \phi(U).$$

$$E : h(a) \text{ :- } p(a,b), p(a,c), p(d,a), r(b,f), o(b,c), q(d,e), t(f,g), \\ \pi(a), \phi(a), \sigma(a), \tau(a), \sigma(b), \tau(b), \phi(b), \tau(d), \rho(d), \pi(f), \phi(f), \sigma(f).$$

3.1 Object Similarity

Consider two clauses C' and C''. Call $A' = \{a'_1, \ldots, a'_n\}$ the set of terms in C', and $A'' = \{a''_1, \ldots, a''_m\}$ the set of terms in C''. When comparing a pair of objects $(a', a'') \in A' \times A''$, two kinds of object features can be distinguished: the properties they own as expressed by unary predicates (*characteristic features*), and the roles they play in n-ary predicates (*relational features*). More precisely, a *role* can be seen as a couple $R = (predicate, position)$ (written compactly as $R = predicate/arity.position$), since different positions actually refer to different roles played by the objects. For instance, a characteristic feature could be `male(X)`, while relational features in a `parent(X,Y)` predicate are the 'parent' role (*parent*/2.1) the 'child' role (*parent*/2.2).

Two corresponding similarity values can be associated to a' and a'': a *characteristic similarity*,

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

based on the set P' of properties related to a' and the set P'' of properties related to a'', for the following parameters:

$n_c = |P' \setminus P''|$ number of properties owned by a' in C' but not by a'' in C'' (*characteristic residual* of a' wrt a'');
$l_c = |P' \cap P''|$ 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' (*characteristic residual* of a'' wrt a').

and a *relational similarity*,

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

based on the *multi*sets R' and R'' of roles played by a' and a'', respectively, for the following parameters:

Table 1. Object features

C			E		
t'	P'	R'	t''	P''	R''
X	$\{\pi, \phi, \rho\}$	$\{p/2.1, p/2.1, p/2.2\}$	a	$\{\pi, \phi, \sigma, \tau\}$	$\{p/2.1, p/2.1, p/2.2\}$
Y	$\{\pi, \sigma, \tau\}$	$\{p/2.2, r/2.1, o/2.1\}$	b	$\{\sigma, \tau\}$	$\{p/2.2, r/2.1, o/2.1\}$
Z	$\{\phi\}$	$\{p/2.2, o/2.2\}$	c	$\{\phi\}$	$\{p/2.2, o/2.2\}$
W	$\{\sigma, \tau\}$	$\{p/2.1, p/2.1, p/2.2\}$	d	$\{\tau, \rho\}$	$\{p/2.1, p/2.1\}$
U	$\{\pi, \phi\}$	$\{r/2.2, s/2.1\}$	f	$\{\pi, \phi, \sigma\}$	$\{r/2.2, t/2.1\}$

$n_r = |R' \setminus R''|$ how many times a' plays in C' role(s) that a'' does not play in C'' (*relational residual* of a' wrt a'');
$l_r = |R' \cap R''|$ 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' (*relational residual* of a'' wrt a').

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

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

Example 2. The properties and roles for some terms in C and E, and the comparison for some of the possible pairs, are reported in Table 1.

3.2 Structural Similarity

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. The structure of a formula is defined by the way in which n-ary *atoms* (predicates applied to a number of terms equal to their arity) are applied to the various objects to relate them. 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 correspond to different terms in the other formula).

Given an n-ary literal, we define its *star* as the multiset of n-ary predicates corresponding to the literals linked to it by some common term (a predicate can appear in multiple instantiations among these literals). The *star similarity* between two compatible n-ary literals l' and l'' having stars S' and S'', respectively, can be computed for the following parameters:

$n_s = |S' \setminus S''|$ how many more relations l' has in C' than l'' has in C'' (*star residual* of l' wrt l'');
$l_s = |S' \cap S''|$ number of relations that both l' in C' and l'' in C'' have in common;

$m_s = |S'' \setminus S'|$ how many more relations l'' has in C'' than l' has in C' (*star residual* of l'' wrt l').

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:

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

where C^s is a composition function (e.g., the average).

Then, any first-order logic formula can be represented as a graph in which atoms are the nodes, and edges connect two nodes *iff* they are related in some way. Leveraging on the fact that clauses are made up by just a single atom in the head and a conjunction of atoms in the body, we can exploit a graph representation that is easier than that for general formulæ, as described in the following. In particular, we will deal with *linked* clauses only (i.e. clauses whose associated graph is connected), and will build the graph based on a term sharing between couples of atoms. 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 yields a Directed Acyclic Graph (DAG), *stratified* in such a way that the head is the only node at level 0 and each successive level is made up by nodes not yet reached by edges that have at least one term in common with nodes in the previous level. In particular, 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.

Example 3. In the graph G_C, the head represents the 0-level of the stratification. Then directed edges may be introduced from $h(X)$ to $p(X, Y)$, $p(X, Z)$ and $p(W, X)$, which yields level 1 of the stratification. Now the next level can be built, adding directed edges from atoms in level 1 to the atoms not yet considered that share a variable with them: $r(Y, U)$ – end of an edge starting from $p(X, Y)$ –, $o(Y, Z)$ – end of edges starting from $p(X, Y)$ and $p(X, Z)$ – and $q(W, W)$ – end of an edge starting from $p(W, X)$. The third level of the graph includes the only remaining atom, $s(U, V)$ – having an incoming edge from $r(Y, U)$.

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. Being such paths univoquely determined reduces the amount of indeterminacy in the comparison. Given two clauses C' and C'', we define the *intersection* between two paths $p' =< l'_1, \ldots, l'_{n'} >$ in $G_{C'}$ and $p'' =< l''_1, \ldots, l''_{n''} >$ in $G_{C''}$ as the pair of longest compatible initial subsequences of p' and p'':

Table 2. Structure features

C		E	
l'	S'	l''	S''
$p(X,Y)$	$\{p/2, p/2, r/2, o/2\}$	$p(a,b)$	$\{p/2, p/2, r/2, o/2\}$
$p(X,Z)$	$\{p/2, p/2, o/2\}$	$p(a,c)$	$\{p/2, p/2, o/2\}$
$r(Y,U)$	$\{p/2, o/2, s/2\}$	$r(b,f)$	$\{p/2, o/2, t/2\}$

Path No.	C	E
1.	$< p(X,Y), r(Y,U), s(U,V) >$	$< p(a,b), r(b,f), t(f,g) >$
2.	$< p(X,Y), o(Y,Z) >$	$< p(a,b), o(b,c) >$
3.	$< p(X,Z), o(Y,Z) >$	$< p(a,c), o(b,c) >$
4.	$< p(W,X), q(W,W) >$	$< p(d,a), q(d,e) >$

$p' \cap p'' = (p_1, p_2) = (< l'_1, \ldots, l'_k >, < l''_1, \ldots, l''_k >)$ 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 $l''_1, \ldots, l''_{k+1})$
and the two residuals as the incompatible trailing parts:
$p' \setminus p'' =< l'_{k+1}, \ldots, l'_{n'} >, p'' \setminus p' =< l''_{k+1}, \ldots, l''_{n''} >)$

Hence, the *path similarity* between p' and p'', $\mathrm{sf}_s(p', p'')$, can be 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'' (*path residual* 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' (*path residual* of p'' wrt p').

by taking into account also the star similarity values for all pairs of literals associated by the initial compatible sequences:

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

where C^p is a composition function (e.g., the average).

Example 4. Since the head is unique (and hence can be uniquely matched), in the following we will deal only with the body literals for structural criteria. Table 2 reports the stars and paths of literals in C and E.

Note that no single criterion is by itself neatly discriminant, but their cooperation succeeds in distributing the similarity values and in making the difference ever clearer as long as they are composed one ontop the previous ones.

3.3 Clause Similarity

Now, similarity between two clauses with the same head predicate can be computed according to their generalization. In particular, one would like to exploit their *least general generalization*, 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 least general generalization [14], at the expenses of some unpleasant side-effects concerning the need of computing its reduced equivalent (and also of some counter-intuitive aspects of the result), or, as most ILP learners do, 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 [6], based on the Object Identity assumption, 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 least general generalization, 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 literals:

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

by taking into account also the star similarity values for all pairs of literals 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). This function evaluates the similarity of two clauses according to the composite similarity of a maximal subset of their literals 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 literals and objects between the two (whose weight in the final evaluation should not overwhelm the similarity coming from the detailed comparisons, hence the multiplication).

3.4 Similarity-guided Clause Generalization

In particular, the similarity formula itself can be exploited for computing the generalization. The path intersections are considered by decreasing similarity,

adding to the partial generalization generated thus far the common literals of each pair whenever they are compatible [7].

The question arises whether the proposed similarity framework is actually able to lead towards the identification of the proper sub-parts to be put in correspondence in the two descriptions under comparison. Since the 'correct' association is a semantic-related issue, this can be evaluated only indirectly as the portion of literals in the clauses to be generalized that is preserved by the generalization: the more literals the generalization preserves from the clauses, the less general it is. More formally, the compression factor (computed as the ratio between the length of the generalization and that of the shortest clause to be generalized) should be as high as possible. Interestingly, on the document dataset (see section 5 for details) the similarity-driven generalization preserved on average more than 90% literals of the shortest clause, with a maximum of 99,48% (193 literals out of 194, against an example of 247) and just 0,006 variance. As a consequence, one woud expect that the produced generalizations are least general ones or nearly so. Noteworthly, using the similarity function on the document labelling task leads to runtime savings that range from 1/3 up to 1/2, in the order of hours.

4 Related Works

Few works faced the definition of similarity or distance measures for first-order descriptions. [5] proposes a distance measure based on probability theory applied to the formula components. Compared to that, our function does not require the assumptions and simplifying hypotheses to ease the probability handling, and no *a-priori* knowledge of the representation language is required. 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 [5].

KGB [2] uses a similarity function, parameterized by the user, to guide generalization; our approach is more straightforward, and can be easily extended to handle negative information in the clauses. In *RIBL* [4] object similarity depends on the similarity of their attributes' values and, recursively, on the similarity of the objects related to them, which poses the problem of indeterminacy.

[18] presents an approach for the induction of a distance on FOL examples, that exploits the truth values of whether each clause covers the example or not as features for a distance on the space $\{0,1\}^k$ between the examples. [13] organizes terms in an importance-related hierarchy, and proposes a distance between terms based on interpretations and a level mapping function that maps every simple expression on a natural number. [15] presents a distance function between atoms based on the difference with their lgg, and uses it to compute distances between clauses. It consists of a pair where the second component allows to differentiate cases where the first component cannot.

5 Experiments on Clustering

Cluster analysis concerns the organization of 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. The variety of techniques for representing data, similarity between elements and strategies to group elements has produced a rich assortment of clustering methods (see [10] for a survey). The classical strategies can be divided in bottom-up and top-down. In the former, each element of the dataset is considered as a cluster. Successively, the algorithm tries to group the clusters that are more similar according to the similarity measure. This step is performed until the number of clusters the user requires as a final result is reached, or the minimal similarity value among clusters is greater than a given threshold. In the latter approach, known as hierarchical clustering, at the beginning all the elements of the dataset form a unique cluster. Successively, the cluster is partitioned into clusters made up of elements that are more similar according to the similarity measure. This step is performed until the number of clusters required by the user as a final result is reached. A further classification is based on whether an element can be assigned (NotExclusive or Fuzzy Clustering) or not (Exclusive or Hard Clustering) to more than one cluster. Also the strategy exploited to partition the space is a criterion used to classify the clustering techniques: in Partitive Clustering a representative point (centroid, medoid, etc..) of the cluster in the space is chosen; Hierarchical Clustering produces a nested series of partitions by merging (Hierarchical Agglomerative) or splitting (Hierarchical Divisive) clusters, Density-based Clustering considers the density of the elements around a fixed point.

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. However, most of the clustering algorithms and systems work on attribute-value representation (e.g., `CLUSTER/2` [12], `CLASSIT` [9], `COBWEB` [8]). Other systems such as `LABYRINTH` [19] can deal with structured objects exploiting a representation that is not powerful enough to express the dataset in a lot of domains. There are few systems that cluster examples represented in FOL (e.g., `AUTOCLASS-like` [16], `KBG` [1]), some of which still rely on propositional distance measures (e.g., `TIC` [3]).

The proposed similarity framework was tested on the conceptual clustering task, where a set of items must be grouped into homogeneous classes according to the similarity between their first-order logic description. In particular, we adopted the classical K-means clustering technique. However, since first-

order logic formulæ do not induce a heuclidean space, it was not possible to identify/build a *centroid* prototype for the various clusters according to which performing the next distribution in the loop. For this reason, we based the distribution on the concept of *medoid* prototypes, where a medoid is defined as the observation that actually belongs to a cluster and that has the minimum average distance from all the other members of the cluster. As to the stop criterion, it was set as the moment in which a new iteration outputs a partition already seen in previous iterations. Note that it is different than performing the same check on the set of prototypes, since different prototypes could yield the same partition, while there cannot be several different sets of prototypes for one given partition. In particular, it can happen that the last partition is the same as the last-but-one, in which case a fixed point is reached and hence a single solution has been found and has to be evaluated. Conversely, when the last partition equals a previous partition, but not the last-but-one one, a loop is identified, and one cannot focus on a single minimum to be evaluated.

Experiments on Conceptual Clustering were run on a real-world dataset[1] containing 353 descriptions of scientific papers first page layout, belonging to 4 different classes: Elsevier journals, Springer-Verlag Lecture Notes series (SVLN), Journal of Machine Learning Research (JMLR) and Machine Learning Journal (MLJ). The complexity of such a dataset is considerable, and concerns several aspects of the dataset: 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 literals, for an average of more than 192 literals per description (some descriptions are made up of more than 400 literals); last, the description is heavily based on a *part_of* relation that increases indeterminacy. Since the class of each document in the dataset is known, we performed a supervised clustering: after hiding the correct class to the clustering procedure, we provided it with the 'anonymous' dataset, asking for a partition of 4 clusters. Then, we compared each outcoming cluster with each class, and assigned it to the best-matching class according to precision and recall. In practice, we found that 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.

The clustering procedure was run first on 40 documents randomly selected from the dataset, then on 177 documents and lastly on the whole dataset, in order to evaluate its performance behaviour when takling increasingly large data. Results are reported in Table 3: for each dataset size it reports the number of instances in each cluster and in the corresponding class, the number of matching instances between the two and the consequent precision (Prec) and recall (Rec) values, along with the overall number of correctly split documents in the dataset. Compound statistics, shown below, report the average precision and recall for each dataset size, along with the overall accuracy, plus some information about runtime and number of description comparisons to be carried out.

[1] `http://lacam.di.uniba.it:8000/systems/inthelex/index.htm#datasets`

Table 3. Experimental results

Instances	Cluster	Class	Intersection	Prec (%)	Rec (%)	Total Overlapping
40	8	Elsevier (4)	4	50	100	35
	6	SVLN (6)	5	83,33	83,33	
	8	JMLR (8)	8	100	100	
	18	MLJ (22)	18	100	81,82	
177	30	Elsevier (22)	22	73,33	100	164
	36	SVLN (38)	35	97,22	92,11	
	48	JMLR (45)	45	93,75	100	
	63	MLJ (72)	62	98,41	86,11	
353	65	Elsevier (52)	52	80	100	326
	65	SVLN (75)	64	98,46	85,33	
	105	JMLR (95)	95	90,48	100	
	118	MLJ (131)	115	97,46	87,79	

Instances	Runtime	Comparisons	Avg Runtime (sec)	Prec (%)	Rec (%)	Acc (%)
40	25'24"	780	1,95	83,33	91,33	87,5
177	9h 34' 45"	15576	2,21	90,68	94,56	92,66
353	39h 12' 07"	62128	2,27	91,60	93,28	92,35

Looking at the compound outcomes, it is evident that the proposed method si highly effective since it is able to autonomously recognize the original classes with precision, recall and accuracy well above 80% and, for larger datasets, always above 90%. This is very encouraging, especially in the perspective of the representation-related difficulties. Runtime refers almost completely to the computation of the similarity between all couples of observations: computing each similarity takes on average about 2sec, which is not much considering the descriptions complexity and the fact that the prototype has not been optimized in this preliminary version. The detailed results show that the reduced dataset performs worse, probably due to the lack of sufficient information for properly discriminating the clusters.

6 Conclusions

First-Order Logic suffers from indeterminacy in mapping portions of descriptions onto each other. This paper proposes a new similarity framework (including criteria, a function and composition techniques for similarity assessment) for Horn clauses, and exploits it for Conceptual Clustering. According to the experimental outcomes, the clustering procedure endowed with the proposed similarity tool reaches very high results in terms of accuracy, precision and recall on a difficult real-world domain dataset. Future work will concern fine-tuning of the similarity computation methodology, and a more extensive experimentation.

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.R. Swartout, editor, *Proc. of AAAI-92*, pages 82–87, 1992.

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

[5] 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.

[6] Floriana Esposito, Nicola Fanizzi, Stefano Ferilli, and Giovanni Semeraro. A generalization model based on oi-implication for ideal theory refinement. *Fundam. Inform.*, 47(1-2):15–33, 2001.

[7] S. Ferilli, T.M.A. Basile, N. Di Mauro, M. Biba, and F. Esposito. Similarity-guided clause generalization. In *Proc. of AI*IA-2007*, LNAI, page 12. Springer, 2007 (To appear).

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

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

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

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

[12] 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.

[13] 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.

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

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

[16] 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.

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

[18] 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.

[19] 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.