# Plugging Numeric Similarity in First-Order Logic Horn Clauses Comparison

S. Ferilli[1,2], T.M.A. Basile[1], N. Di Mauro[1,2], and F. Esposito[1,2]

[1] Dipartimento di Informatica, Università di Bari
{ferilli,basile,ndm,esposito}@di.uniba.it
[2] Centro Interdipartimentale per la Logica e sue Applicazioni, Università di Bari

**Abstract.** Horn clause Logic is a powerful representation language exploited in Logic Programming as a computer programming framework and in Inductive Logic Programming as a formalism for expressing examples and learned theories in domains where relations among objects must be expressed to fully capture the relevant information. While the predicates that make up the description language are defined by the knowledge engineer and handled only syntactically by the interpreters, they sometimes express information that can be properly exploited only with reference to a suitable background knowledge in order to capture unexpressed and underlying relationships among the concepts described. This is typical when the representation includes numerical information, such as single values or intervals, for which simple syntactic matching is not sufficient.

This work proposes an extension of an existing framework for similarity assessment between First-Order Logic Horn clauses, that is able to handle numeric information in the descriptions.

The viability of the solution is demonstrated on sample problems.

## 1 Introduction

First-Order Logic (*FOL* for short) is a powerful representation language that allows to express relationships among objects, which is often an unnegligible requirement in real-world and complex domains. Logic Programming [7] is a computer programming framework based on a FOL sub-language, which allows to perform reasoning on knowledge expressed in the form of Horn clauses. Inductive Logic Programming (ILP) [8] aims at learning automatically logic programs from known examples of behaviour, and has proven to be a successful Machine Learning approach in domains where relations among objects must be expressed to fully capture the relevant information. Many AI tasks can take advantage from techniques for descriptions comparison: subsumption procedures (to converge more quickly), flexible matching, instance-based classification techniques or clustering, generalization procedures (to focus on the components that are more likely to correspond to each other). In FOL, this is a particularly complex task due to the problem of *indeterminacy* in mapping portions of one formula onto portions of another.

R. Pirrone and F. Sorbello (Eds.): AI*IA 2011, LNAI 6934, pp. 33–44, 2011.

Usually, predicates that make up the description language used to tackle a specific problem are defined by the knowledge engineer that is in charge of setting up the reasoning or learning task, and are handled as purely syntactic entities by the systems. However, the use of uninterpreted predicates and terms (meaning by 'interpretation' their mapping onto meaningful objects, concepts and relationships) is often too limiting for an effective application of this kind of techniques to real-world problems. Indeed, in the real world there are a huge number of implicit connections and inter-relationships between items that would be ignored by the system. For limited and simple domains, only a few of these relationships are actually significant, and must be expressed not to prevent finding a solution. In these cases, they can be provided in the form of a background knowledge. However, if the amount of relevant information to be expressed as background knowledge grows, this becomes infeasible manually and requires the support of readily available resources in the form of explicit knowledge items or computational procedures.

This work builds on previous results concerning a framework for similarity assessment between FOL Horn clauses, where the overall similarity depends on the similarity of the pairs of literals associated by the least general generalization, the similarity of two literals in turn depends on the similarity of their corresponding arguments (i.e., terms), and the similarity between two terms is computed according to the predicates and positions in which they appear. Following a previous paper in which the framework was extended to handle taxonomic knowledge, here, a novel and general approach to the assessment of similarity between numeric values and intervals is proposed, and its integration as a corresponding further extension of the similarity framework for clauses including numeric information is described.

The rest of this paper is organized as follows. Section 3 introduces the basic formula and framework for the overall assessment of similarity between Horn clauses. Section 4 proposes an application of the same formula to compute the numeric similarity between values and/or intervals, and introduces it in the previous framework. Section 5 shows experiments that suggest the effectiveness of the proposed approach. Lastly, Section 6 concludes the paper and outlines future work directions.

## 2 Background

Let us preliminary recall some basic notions involved in Logic Programming. The *arity* of a predicate is the number of arguments it takes. A *literal* is an $n$-ary predicate, applied to $n$ terms, possibly negated. *Horn clauses* are logical formulæ usually represented in Prolog style as $l_0 \text{ :- } l_1, \ldots, l_n$ where the $l_i$'s are *literals*. It corresponds to an implication $l_1 \wedge \cdots \wedge l_n \Rightarrow l_0$ 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". Datalog [2] is, at least syntactically, a restriction of Prolog in which, without loss of generality [9], only variables and constants (i.e., no functions) are allowed as terms. A set of literals is *linked* if

and only if each literal in the set has at least one term in common with another literal in the set. We will deal with the case of linked Datalog clauses. In the following, we will call *compatible* two sets or sequences of literals 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).

Real-world problems, and the corresponding descriptions, often involve numeric features, that are to be expressed in the problem formalization and handled by the inferential procedures. For instance, when describing a bicycle we would like to say that the front wheel diameter is 28 inches, or when defining the title block in a scientific paper we would like to say that it must be placed in a range going from 5% to 20% of the page height from the top. Let us call this kind of features (such as size, height in the above examples) *numeric attributes*. Clearly, to be properly handled such a kind of information needs to be suitably interpreted according to a background knowledge consisting of the mathematical models of numbers and their ordering relationships. Unfortunately, the purely logical setting ignores such a background knowledge, and considers each single value as completely unrelated to all other values. This problem has been tackled in two different ways.

Keeping the purely logical setting ensures general applicability of the logical representation and inference techniques: a classical solution in this case has been discretization of the range of numeric values allowed for a given attribute into pre-defined intervals, each of which can be associated to a symbolic descriptor (e.g., size_small, size_large, ...; position_top, position_middle, ...). Thus, the original descriptions are to be pre-processed to turn all instances of numeric attributes into the corresponding discretized descriptors. What is a useful number of intervals in which splitting the range of values allowed for a numeric attribute? How to choose the cut points between intervals? Both choices are crucial, since once it is determined even points that are very close to each other (e.g., 4.999 and 5.001 for a cut point placed at 5) will be considered as two completely different entities. Techniques for (semi-)automatic definition of the intervals given samples of values for the attribute have been proposed (e.g., [1]), based on statistics on the values occurrence and distribution, although a (partial) manual intervention is often required to provide and/or fix their outcome. In any case, if the intervals are not to be considered as completely distinct entities, the problem is simplified but not completely solved, since additional background knowledge must be provided to express the ordering relationships between intervals (requiring a number of items that is quadratic in the number of intervals, to express which one precedes which other for all possible pairs) or progressive levels of aggregations of groups of adjacent intervals into wider ones (requiring, for all possible combinations, a number of items that is exponential in the number of intervals).

As another option, plugging the ability to handle numeric information directly in the inference engine somehow 'spoils' its behavior and adds complexity (reducing efficiency). A problem (in both cases) is the fact that the specific way in which numeric information is to be handled is strictly domain-dependent: Are

values 15 and 300 close or distant (and how much are they)? This question cannot be answered in general (a difference of, say, 215 meters might be meaningful when comparing two fields, but completely insignificant when comparing planets according to their size.

## 3 Similarity Framework

In this section, a short description of the similarity framework in its current status, borrowed from [5], will be provided. The original framework for computing the similarity between two Datalog Horn clauses has been provided in [4]. Intuitively, the evaluation of similarity between two items $i'$ and $i''$ might be based both on parameters expressing the amounts of common features, which should concur in a positive way to the similarity evaluation, and of the features of each item that are not owned by the other (defined as the *residual* of the former with respect to the latter), which should concur negatively to the whole similarity value assigned to them [6]:

$n$ , 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 similarity function that expresses the degree of similarity between $i'$ and $i''$ based on the above parameters, and that has a better behaviour than other formulæ in the literature in cases in which any of the parameters is 0, is [4]:

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

It takes values in $]0, 1[$, which resembles the theory of probability and hence can help human interpretation of the resulting value. When $n = m = 0$ it tends to the limit of 1 as long as the number of common features grows. The full-similarity value 1 is never reached, being reserved to two items that are exactly the same ($i' = i''$), which can be checked in advance. Consistently 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 is also always strictly greater than 0, and will tend to such a value as long as the number of non-shared features grows. Moreover, for $n = l = m = 0$ the function evaluates to 0.5, which can be considered intuitively correct for a case of maximum uncertainty. Note that each of the two terms refers specifically to one of the two items under comparison, and hence they could be weighted to reflect their importance.

In FOL representations, usually terms denote objects, unary predicates represent object properties and $n$-ary predicates express relationships between objects; hence, the overall similarity must consider and properly mix all such components. The similarity between two clauses $C'$ and $C''$ is guided by the similarity between their structural parts, expressed by the $n$-ary literals in their bodies, and is a function of the number of common and different objects and relationships between them, as provided by their least general generalization

$C = l_0 \text{ :- } l_1, \ldots, l_k$. Specifically, we refer to the $\theta_{OI}$ generalization model [3]. The resulting formula is the following:

$$\text{fs}(C', C'') = \text{sf}(k' - k, k, k'' - k) \cdot \text{sf}(o' - o, o, o'' - o) + \text{avg}(\{\text{sf}_s(l'_i, l''_i)\}_{i=1,\ldots,k})$$

where $k'$ is the number of literals and $o'$ the number of terms in $C'$, $k''$ is the number of literals and $o''$ the number of terms in $C''$, $o$ is the number of terms in $C$ and $l'_i \in C'$ and $l''_i \in C''$ are generalized by $l_i$ for $i = 1, \ldots, k$. The similarity of the literals is smoothed by adding the overall similarity in the number of overlapping and different literals and terms.

The similarity between two compatible $n$-ary literals $l'$ and $l''$, in turn, depends on the multisets of $n$-ary predicates corresponding to the literals directly linked to them (a predicate can appear in multiple instantiations among these literals), called *star*, and on the similarity of their arguments:

$$\text{sf}_s(l', l'') = \text{sf}(n_s, l_s, m_s) + \text{avg}\{\text{sf}_o(t', t'')\}_{t'/t'' \in \theta}$$

where $\theta$ is the set of term associations that map $l'$ onto $l''$ and $S'$ and $S''$ are the stars of $l'$ and $l''$, respectively:

$$n_s = |S' \setminus S''| \qquad l_s = |S' \cap S''| \qquad m_s = |S'' \setminus S'|$$

Lastly, the similarity between two terms $t'$ and $t''$ is computed as follows:

$$\text{sf}_o(t', t'') = \text{sf}(n_c, l_c, m_c) + \text{sf}(n_r, l_r, m_r)$$

where the former component takes into account the sets of properties (unary predicates) $P'$ and $P''$ referred to $t'$ and $t''$, respectively:

$$n_c = |P' \setminus P''| \qquad l_c = |P' \cap P''| \qquad m_c = |P'' \setminus P'|$$

and the latter component takes into account how many times the two objects play the same or different roles in the $n$-ary predicates; in this case, since an object might play the same role in many instances of the same relation, the *multi*sets $R'$ and $R''$ of roles played by $t'$ and $t''$, respectively, are to be considered:

$$n_r = |R' \setminus R''| \qquad l_r = |R' \cap R''| \qquad m_r = |R'' \setminus R'|$$

This general (uninterpreted) framework was extended [5] to consider taxonomic information (assuming that there is some way to distinguish taxonomic predicates from ordinary ones). Using the same similarity function, the parameters for taxonomic similarity between two concepts are determined according to their associated sets of ancestors (say $I_1$ and $I_2$) in a given heterarchy of concepts: their intersection (i.e., the number of common ancestors) is considered as common information (yielding $l_t = |I_1 \cap I_2|$), and the two symmetric differences as residuals (yielding $n_t = |I''_1 - I'_1|$ and $m_t = |I''_2 - I'_2|$). Since the taxonomic predicates represent further information about the objects involved in a description, in addition to their properties and roles, term similarity is the component where the corresponding similarity can be introduced in the overall framework:

$$\text{sf}_o(t', t'') = \text{sf}(n_c, l_c, m_c) + \text{sf}(n_r, l_r, m_r) + \text{sf}(n_t, l_t, m_t)$$

where the additional component refers to the number of common and different ancestors of the two concepts associated to the two terms, as specified above.

## 4 Numeric Similarity

Given the above considerations, it is clear that, in an extended framework considering interpreted predicates and constants in addition to simple syntactic entities, the ability to handle numeric information is at least as fundamental as considering conceptual ones. Hence, the motivation for this work on the extension of the similarity-related technique. While observations usually described specific cases using single constants, models typically refer to allowed ranges of values: thus, we are interested in handling all the following cases:

- comparison between two intervals
- comparison between an interval and a value
- comparison between two values

To ensure a smooth integration of the new components, a first requirement is preserving the same basic similarity function, whence the need to specify how to extract parameters $l$, $n$ and $m$ from numeric comparisons. In this respect, let us start our discussion from the case of comparison between two intervals, say $[i'_1, i'_2]$ and $[i''_1, i''_2]$. Two intuitive approaches are available (assume, without loss of generality, that $i'_1 \leq i'_2$):

- basing the comparison on the distance between the interval **extremes**: then, we take $n = |i''_1 - i'_1|$, $m = |i''_2 - i'_2|$ and $l = \min(i'_2, i''_2) - \max(i'_1, i''_1)$ if non-negative (or 0 otherwise). Note that this solution does not take into account the actual distance between the two intervals when they are disjoint, but modifying the function to take into account this distance as a negative value would spoil uniformity of the approach and make the function definition more complex.
- considering the intervals as **sets**, and exploiting set operators and interpretation: $l$ would be the width (expressed as $||\cdot||$) of the overlapping part ($l = ||[i'_1, i'_2] \cap [i''_1, i''_2]||$), and $n, m$ their symmetric differences, respectively ($n = ||[i'_1, i'_2] \setminus [i''_1, i''_2]||$, $m = ||[i''_1, i''_2] \setminus [i'_1, i'_2]||$).

Both strategies can be straightforwardly applied also to the comparison of an interval to a single value, considered as an interval in which both extremes are equal. However, the $l$ parameter would always be zero in this case.

Let us now check and evaluate the behavior of the two candidate approaches by applying them on a set of sample intervals, as shown in Table 1. Overall, both approaches seem reasonable. As expected, their outcome is the same for partially overlapping intervals, so that case is not a discriminant to prefer either over the other. Different behavior emerges in the cases of disjoint intervals or of inclusion of intervals (which is always the case of an interval compared to a single value). In the former, the extreme-based approach ensures more distinction power, because the distance between the intervals is taken into account. While this behavior seems reasonable (according to the intuition that the farther two intervals, the more different they are), on the other hand, in the case of an interval being a subinterval of the other it is not. Indeed, the set-based approach charges the whole

**Table 1.** Similarity values between sample intervals

| Intervals | | Extreme-based | | | | Set-based | | | |
|---|---|---|---|---|---|---|---|---|---|
| $I_1$ | $I_2$ | $n$ | $l$ | $m$ | similarity | similarity | $n$ | $l$ | $m$ |
| $[10, 15]$ | $[11, 16]$ | 1 | 4 | 1 | $0.\overline{714285}$ | $0.\overline{714285}$ | 1 | 4 | 1 |
| $[10, 15]$ | $[10, 15]$ | 0 | 5 | 0 | $0.\overline{857142}$ | $0.\overline{857142}$ | 0 | 5 | 0 |
| $[21, 25]$ | $[1, 5]$ | 20 | 0 | 20 | $0.0\overline{45}$ | $0.1\overline{6}$ | 4 | 0 | 4 |
| $[1, 5]$ | $[21, 25]$ | 20 | 0 | 20 | $0.0\overline{45}$ | $0.1\overline{6}$ | 4 | 0 | 4 |
| $[1, 5]$ | $[6, 10]$ | 5 | 0 | 5 | $0.\overline{142857}$ | $0.1\overline{6}$ | 4 | 0 | 4 |
| $[1, 5]$ | $[0, 6]$ | 1 | 4 | 1 | $0.\overline{714285}$ | $0.7291\overline{6}$ | 0 | 4 | 2 |

**Table 2.** Similarity values between sample interval-value pairs

| Intervals | | Extreme-based | | | | Set-based | | | |
|---|---|---|---|---|---|---|---|---|---|
| $I$ | $v$ | $n$ | $l$ | $m$ | similarity | similarity | $n$ | $l$ | $m$ |
| $[1, 5]$ | $1 \equiv [1, 1]$ | 0 | 0 | 4 | $0.\overline{3}$ | $0.\overline{3}$ | 4 | 0 | 0 |
| $[1, 5]$ | 2 | 1 | 0 | 3 | $0.2\overline{6}$ | $0.\overline{3}$ | 4 | 0 | 0 |
| $[1, 5]$ | 3 | 2 | 0 | 2 | 0.25 | $0.\overline{3}$ | 4 | 0 | 0 |
| $[1, 5]$ | 4 | 3 | 0 | 1 | $0.2\overline{6}$ | $0.\overline{3}$ | 4 | 0 | 0 |
| $[1, 5]$ | 5 | 4 | 0 | 0 | $0.\overline{3}$ | $0.\overline{3}$ | 4 | 0 | 0 |
| $[1, 5]$ | $6 \equiv [6, 6]$ | 5 | 0 | 1 | $0.\overline{238095}$ | $0.\overline{3}$ | 0 | 0 | 4 |
| $[1, 5]$ | 21 | 20 | 0 | 16 | $0.0\overline{5}$ | $0.\overline{3}$ | 4 | 0 | 0 |
| $[7, 10]$ | 11 | 4 | 0 | 1 | 0.25 | 0.35 | 3 | 0 | 0 |
| $[7, 10]$ | 14 | 7 | 0 | 4 | $0.13\overline{8}$ | 0.35 | 3 | 0 | 0 |
| $[7, 10]$ | 3 | 4 | 0 | 7 | $0.13\overline{8}$ | 0.35 | 3 | 0 | 0 |
| $6 \equiv [6, 6]$ | $10 \equiv [10, 10]$ | 4 | 0 | 4 | $0.1\overline{6}$ | 0.5 | 0 | 0 | 0 |
| $10 \equiv [10, 10]$ | $10 \equiv [10, 10]$ | 0 | 0 | 0 | 0.5 | 0.5 | 0 | 0 | 0 |

difference to the residual of the larger interval, which complies with the intuition that it has more 'different stuff' that the other does not have; conversely, the extreme-based approach splits such a difference on both parameters $n$ and $m$, resulting in a smaller similarity value.

This is even more evident looking at the behavior of the two approaches in the case of an interval and a single value, as shown in Table 2. Only the case where the first item is an interval and the second one is a value is reported, due to the similarity function being symmetric providing the same result also in the opposite case. Here, given a value included in an interval, their similarity according to the set-based approach is constant, while in the extreme-based approach it is affected by the position of the former within the latter: the closer the value to the middle of the interval, the smaller their similarity; conversely, as long as the value approaches the interval extremes, their similarity grows up to the same similarity as the set-based approach. If we assume that the interval just specifies an allowed range, with no reason to prefer particular regions within that range, the set-based approach is clearly more intuitive. Again, in the case of a value outside the interval (corresponding to disjoint intervals) an opposite evaluation holds: the actual distance of the value from the interval is considered by the extreme-based approach, affecting its evaluation, but not by the set-based

**Table 3.** Similarity values between sample pairs of values

| Values | | Extreme-based | | | | Specific |
|---|---|---|---|---|---|---|
| $v_1$ | $v_2$ | $n$ | $l$ | $m$ | similarity | similarity |
| 1 | 1 | 0 | 0 | 0 | 0.5 | 1 |
| 1 | 2 | 1 | 0 | 1 | $0.\overline{3}$ | 0.5 |
| 1 | 3 | 2 | 0 | 2 | 0.25 | $0.\overline{3}$ |
| 1 | 4 | 3 | 0 | 3 | 0.2 | 0.25 |
| 1 | 5 | 4 | 0 | 4 | $0.1\overline{6}$ | 0.2 |
| 1 | 10000 | 9999 | 0 | 9999 | $0.\overline{00009999}$ | 0.0001 |
| 6 | 4 | 2 | 0 | 2 | 0.25 | $0.\overline{3}$ |

approach, where the absurd that a value outside a range has a larger similarity than a value falling in the range happens. In both approaches, the larger the interval, the smaller the similarity (which is consistent with the intuition that a value is more likely to fall in a wider range than in a narrower one).

When comparing two values, in particular, the set-based approach returns maximum uncertainty about their similarity (0.5) due to all parameters being zero, and hence it is not applicable. The extreme-based approach evaluates their similarity according to how close to each other they are on the real-valued axis, but loses expressive power (because any pair of values yields $n = m$), and has the additional drawback that when comparing a value to itself it yields $n = l = m = 0$ and hence similarity 0.5 (whereas we would expect to get 1 as a perfect matching). Thus, a different approach is needed. The similarity assessment should be independent of the different ranges of values used in the specific domain (e.g., the range for describing the length of a pen is incomparable to that for describing the width of a building). We propose the following formula:

$$\mathrm{sf}_n(v_1, v_2) = \frac{1}{|v_1 - v_2| + 1}$$

Let us briefly examine the behavior of such a function. It is clearly symmetric. When the difference between the two values approaches zero it approaches 1, and becomes actually 1 for $v_1 = v_2$, as expected (differently from the previous cases in the logical setting, one is sure that two equal values denote exactly the same entity). As long as the difference increases, the function monotonically approaches 0, but never reaches that value (according to the intuition that a larger difference can be always thought of, requiring a smaller similarity value). The rate at which 0 is approached decreases as long as the difference takes larger and larger values, consistently with the intuition that for very large distances one does not care small variations. Of course, if the descriptions are consistent, only values referred to the same kind of entities/attributes will be compared to each other, and hence the corresponding ranges of similarities should be consistent and comparable to each other.

As to the similarity between values, some sample comparisons are reported in Table 3 (both the specific strategy and the extreme-based one are symmetric).

As desired, identity of values yields similarity 1, and wider distances among the two values result in smaller similarity values (independently of the actual values).

Summing up, a specific strategy is needed when comparing two values. When at least an interval is involved, both the set-based and the extreme-based strategies are equivalent in the case of partially overlapping intervals. Otherwise, the former is better in the case of an interval or value being included in another interval, because it better fits the concept on which the similarity function parameters are based. Conversely, the extreme-based strategy is able to consider the actual distance from the interval and/or value extremes in the case of disjoint intervals, which affects the residual parameters. Overall, a cooperation of the three strategies is desirable to fully match the spirit of the similarity function parameters. In this case, a deeper empirical study is adviceable, and is planned as future work, to establish if and how a smooth combination thereof can be obtained, ensuring meaningful overall results and comparable similarity assessments even when determined by different approaches (e.g., the similarity for two distinct values should not be larger than the similarity between a value and an interval it belongs to).

A final issue is where to embed the new similarity component in the overall First-Order Logic similarity framework. Without loss of generality, we can assume that numeric attributes (those associating a numeric value to a given entity) apply to single terms in the overall description, and thus are represented by binary predicates $a(t, v)$ meaning that "object $t$ has (numeric) value $v$ for attribute $a$". Indeed, the case of relationships associated with numeric information (e.g., the weight of arcs in a graph), can be easily handled by reifying the relationship in a term and then associating the value to the new term: e.g., $arc\_weight(n_1, n_2, v)$ would become $arc(n_1, n_2, a), weight(a, v)$. In this setting, the numeric predicates represent further information about the objects involved in a description, in addition to their properties and roles (and taxonomic position, in case), and hence term similarity is the component where the corresponding similarity can be introduced in the overall framework.

Of course, we assume that there is some way to distinguish numeric predicates from ordinary ones, so that they can be handled separately by the procedures (the numeric values themselves should be sufficient for this). The overall similarity between two terms becomes:

$$\mathrm{sf}_o(t', t'') = \mathrm{sf}(n_c, l_c, m_c) + \mathrm{sf}(n_r, l_r, m_r)[+\mathrm{sf}(n_t, l_t, m_t)] + \mathrm{sf}_n(N_1, N_2)$$

where the components can be weighted differently if needed, and the additional component refers to the numeric similarity between intervals and/or single values, as applicable, specified above.

## 5 Evaluation

To fully evaluate the effectiveness of the proposed approach to numeric similarity embedded in the wider First-Order Logic Horn-clause similarity framework, a dataset involving numeric attributes and including both intervals and specific

values should be exploited. Unfortunately, the available datasets typically fill each numeric attribute with a single number expressing its value for the object being described, while intervals are usually exploited in general models rather than in observations. Thus, in this work we will evaluate only the comparison of specific values embedded in the overall structural similarity. We are currently working at building datasets including both intervals and specific values, and an extended evaluation of the full-fledged numeric similarity strategy is planned as future work. Specifically, we focus on the classical problem of Mutagenesis [10], as a most famous success in ILP and as a dataset involving both relational and numeric descriptors. It should be noted that this is a very stressing dataset for our technique, because it is known for being a problem where the key to success is in the relational part, rather than in the attribute/value (numeric) one. Indeed, it was exploited to demonstrate that ILP can learn predictive theories from a dataset on which classical attribute-value techniques based on regression failed. Thus, the numeric values are more likely to act as noise in the proper similarity assessment, rather than as useful information to be leveraged (in other word, one might expect that ignoring numeric information would improve effectiveness). It aims at learning when a chemical compound is active, and when it is not, with respect to mutagenicity. Molecules in this dataset are represented according to the atoms that make them up, and to the bonds linking those atoms, so that the 3D structure of the molecule is completely described. Atom substances are represented by symbolic predicates, while numeric features are used for atom weights and bond charges. The dataset includes 188 examples, of which 63 are positive (class *active*) and 125 are negative (class *not_active*).

The clustering procedure was stopped as soon as a loop was detected: after a few steps, the k-means algorithm started oscillating between two sets of clusters such that applying a further distribution on the former yields the latter, and *vice versa*. Thus, both can be considered as candidate solutions, there being no reason to prefer either over the other. The purity for the two candidate clusters was, however, so close that either choice might be adopted: 76.59% and 76.06%, respectively.

Another experiment, more suitable for evaluation of the correctness of the proposed approach, concerns the problem of classification of documents according to their layout description. In this case, the same dataset as in [3] was exploited, made up of 353 examples of scientific papers from 4 different series: Elsevier journals, Springer-Verlag Lecture Notes (SVLN), Machine Learning Journal (MLJ), Journal of Machine Learning Research (JMLR). The descriptions involve both relational descriptors (for the mutual position and alignment among layout blocks) and attribute-value descriptors for each layout block. In particular, 4 numeric attributes are present: horizontal/vertical position in the page of the block, and width/height thereof. Thus, differently from the Mutagenesis dataset, the expectation is that these descriptors are very significant to class discrimination. Previous applications on this dataset [3] were carried out by discretizing the allowed values for these descriptors into (manually-defined) intervals, and assigning a symbolic descriptor to each such interval. Here, the aim is checking whether

**Table 4.** Dispersion matrix for the Document Clustering problem

| | Elsevier | MLJ | JMLR | SVLN | Total | Errors | Accuracy |
|---|---|---|---|---|---|---|---|
| Cluster 1 | 50 | 1 | 0 | 1 | 52 | 2 | 96.15 |
| Cluster 2 | 7 | 84 | 1 | 0 | 92 | 8 | 91.30 |
| Cluster 3 | 0 | 30 | 99 | 0 | 129 | 30 | 76.74 |
| Cluster 4 | 4 | 7 | 0 | 69 | 80 | 11 | 86.25 |
| Total | 61 | 122 | 100 | 70 | 353 | 51 | 85.55 |
| Missed | 11 | 38 | 1 | 1 | – | – | – |

the introduction of the numeric similarity component, able to handle directly the original observations (and hence avoiding the need for human intervention to provide a discretization knowledge), can provide effective results. Again, a k-means algorithm is run, asked to find 4 groups in the observations obtained by hiding the class information from the above examples. After 100656.767 seconds needed to compute the similarity matrix among all pairs of observations, the resulting clusters are shown in Table 4.

It clearly emerges which clusters represent which classes, due to a predominance of the corresponding elements: Cluster 1 corresponds to Elsevier, including 50 out of its 61 correct elements (plus 2 wrong elements from other classes), Cluster 2 corresponds to MLJ, Cluster 3 corresponds to JMLR and Cluster 4 to SVLN. Given this correspondence, the *purity* of each cluster with respect to the associated class can be computed, as the ratio of elements from that class over the total elements in the cluster. There are 51 errors overall, yielding an overall 85.55% accuracy, that increases to 87.61% taking the average accuracy of the various classes/clusters. Compared to the 92.35% purity reported in [4] it can be considered a satisfactory result, considering that here no help is provided to the system, while there a manual discretization carried out by experts was provided to turn numeric values into symbolic ones (the reference value of *supervised* learning on the same dataset, using the experts' discretization, is 98% accuracy, that can be considered as the counterpart of purity). The worst performing class is MLJ, that is also the largest one however. It has the largest number of missed items (most of which fall in the JMLR class/cluster, and all clusters include at least one element from this class. Indeed, by observing its layout, it turns out that it is in some way at the crossing of the other classes, and in particular the 30 documents in JMLR are actually very similar to real JMLR ones (the Authors blocks are in the same place, under the title, both have a heading at the top of the page — although it is narrower in JMLR). This suggests that these kinds of blocks are the most significant to discriminate different classes in this dataset.

## 6 Conclusions

Horn clause Logic is a powerful representation language for automated learning and reasoning in domains where relations among objects must be expressed to fully capture the relevant information. While the predicates in the description

language are defined by the knowledge engineer and handled only syntactically by the interpreters, they sometimes express information that can be properly exploited only with reference to a background knowledge in order to capture unexpressed and underlying relationships among the concepts described. After a previous work aimed at extending a general similarity framework for First-Order Logic Horn clauses with the ability to deal with taxonomic information, in this paper we presented another extension concerning numeric descriptors involving intervals and/or numeric values. A composite approach to the numeric similarity assessment, compliant with the general framework, has been proposed, discussed and evaluated. Clustering experiments on a typical dataset mixing relational and numeric information show that the proposal is effective.

Future work will concern deeper empirical evaluation of the behavior of the proposed approaches to numeric computation in the case of intervals, and of its integration in the general similarity framework (e.g., to determine what weight it should have with respect to the other similarity parameters). Then, experiments aimed at application of the framework to other real-world problems are planned. Finally, another research direction concerns the exploitation of the proposed technique as a support to refinement operators for incremental ILP systems.

## References

[1] Biba, M., Esposito, F., Ferilli, S., Di Mauro, N., Basile, T.M.A.: Unsupervised discretization using kernel density estimation. In: IJCAI 2007, pp. 696–701 (2007)
[2] Ceri, S., Gottlöb, G., Tanca, L.: Logic Programming and Databases. Springer, Heidelberg (1990)
[3] Esposito, F., Fanizzi, N., Ferilli, S., Semeraro, G.: A generalization model based on oi-implication for ideal theory refinement. Fundamenta Informaticæ 47(1-2), 15–33 (2001)
[4] Ferilli, S., Basile, T.M.A., Biba, M., Di Mauro, N., Esposito, F.: A general similarity framework for horn clause logic. Fundamenta Informaticæ 90(1-2), 43–46 (2009)
[5] Ferilli, S., Biba, M., Mauro, N., Basile, T.M., Esposito, F.: Plugging taxonomic similarity in first-order logic horn clauses comparison. In: Serra, R., Cucchiara, R. (eds.) AI*IA 2009. LNCS, vol. 5883, pp. 131–140. Springer, Heidelberg (2009)
[6] Lin, D.: An information-theoretic definition of similarity. In: Proc. 15th International Conf. on Machine Learning, pp. 296–304. Morgan Kaufmann, San Francisco (1998)
[7] Lloyd, J.W.: Foundations of Logic Programming, 2nd edn. Springer, Heidelberg (1987)
[8] Muggleton, S.: Inductive logic programming. New Generation Computing 8(4), 295–318 (1991)
[9] Rouveirol, C.: Extensions of inversion of resolution applied to theory completion. In: Inductive Logic Programming, pp. 64–90. Academic Press, London (1992)
[10] Srinivasan, A., Muggleton, S., King, R., Sternberg, M.: Mutagenesis: ILP experiments in a non-determinate biological domain. In: Wrobel, S. (ed.) Proceedings of the 4th International Workshop on Inductive Logic Programming. GMD-Studien, vol. 237, pp. 217–232 (1994)