Two Extensions of FOL Horn Clauses Comparison to Interpreted Predicates

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

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

Abstract. First-Order Logic Horn clauses are a powerful representation formalism for domains where relations among objects must be expressed to fully capture the relevant information. While the predicates that make up the description language are handled only syntactically by the interpreters, they sometimes express information that can be properly exploited only with reference to a specific background knowledge in order to capture unexpressed and underlying relationships. Two prototypical examples are taxonomic information (e.g., coming from the words found in a text) and numerical information (e.g., coming from measurements and acceptable ranges), 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 for these two cases. 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 fundamental requirement in real-world and complex domains. Logic Programming [10] 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) [12] aims at learning automatically logic programs from known examples of behavior, 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. In FOL, this is a particularly complex task due to the problem of *indeterminacy* in mapping portions of one formula onto portions of another.

The predicates that make up the description language are usually defined by the knowledge engineer, and are handled as purely syntactic (uninterpreted) entities by the systems. The knowledge engineer can also provide a background knowledge to be exploited in order to improve performance or effectiveness of the results. 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, where 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 paper builds on an existing (uninterpreted) framework for similarity assessment between FOL Horn clauses, and extends it with novel and general approaches to handle two particular kinds of information that often need to be expressed in the descriptions: taxonomic information, conveying implicit relationships among the concepts described, and numeric one, involving single values and/or intervals. The next Section introduces the basic formula and framework for the overall assessment of similarity between Horn clauses. Section 3 proposes a solution to compute the taxonomic similarity between two concepts or words, and shows sample experiments. Section 4 does the same for numeric similarity. Section 5 presents experiments on the effectiveness of introducing the new similarity components in the overall First-Order Logic framework. Lastly, Section 6 concludes the paper and outlines future work directions.

2 Background

The framework for computing the similarity between two Datalog [3] clauses proposed in [5] builds on a function that evaluates the similarity between two items i' and i'' as:

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

It exploits both information that is common to the two items, which increases similarity, and information of each item that are not owned by the other (the *residual* of the former with respect to the latter), which decreases similarity [9]:

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').

It takes values in $]0, 1[$, which resembles the theory of probability and hence can help human interpretation of the resulting value. It has a better behaviour than other formulæ in the literature in cases in which any of the parameters is 0. When $n = m = 0$ it approaches 1 as long as l (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 approach such a value as long as the number of non-shared features grows. For $n = l = m = 0$ it evaluates to 0.5, intuitively associated to 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 differently according to 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 $C = l_0$:- $l_1, \ldots, l_k$. Specifically, we refer to the θ_{OI} generalization model [4]. The resulting formula is the following:

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

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:

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

where θ 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:

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

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 (a *role* being an argument position in a predicate); 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:

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

Since we aim at extending this general similarity framework by considering various kinds of interpreted information, for compatibility and smooth integration in the following we will exploit the same function (1).

3 Taxonomic Similarity Approach

A lot of research has been devoted to develop and test similarity measures for concepts in a taxonomy (a survey for WordNet can be found in [2]). The most exploited relationship is generalization/specialization, relating concepts to their super-/sub-concepts. Many proposals are based on the length of the paths that link the concepts to be compared to their closest common ancestor, according to the intuition that, the closer such an ancestor, the more they can be considered as similar to each other. While this might work in hierarchical taxonomies, where the uniqueness of such an ancestor for any two elements is guaranteed, taxonomies in real-world domains are often heterarchies, where by multiple inheritance a concept can specialize many other concepts. Hence, many incomparable common ancestors and paths between concepts can be found, and going to the single common one would very often result in overgeneralization. Our novel solution to compute the similarity between two concepts c' and c'' in a taxonomy takes into account their whole set of ancestors in the heterarchy (say I' and I'', respectively), and applies (1) by using their intersection (i.e., the number of common ancestors) as common information (yielding $l_a = |I' \cap I''|$), and the two symmetric differences as residuals (yielding $n_a = |I'' \setminus I'|$ and $m_a = |I'' \setminus I'|$)[1]:

$$\mathrm{sf}_t(t', t'') = \mathrm{sf}(n_a, l_a, m_a)$$

Again this is intuitive, since the number of common ancestors can be considered a good indicator of the shared features between the two concepts, just as the number of different ancestors can provide a reasonable estimation of the different information and features they own[2].

Consider now two natural language words w' and w'' associated, respectively, to the sets of concepts C' and C'' in the taxonomy (due to the problem of polysemy, a word may express many concepts). Their similarity assessment must somehow combine the similarities between each pair of concepts underlying them (e.g., taking the average or maximum similarity among such pairs, or exploiting the domain of discourse). We propose the following strategy:

$$\mathrm{sf}_t(w', w'') = \max_{c' \in C', c'' \in C''} \mathrm{sf}(c', c'')$$

[1] In hierarchical taxonomies, that are tree-shaped, the path connecting any node (concept) to the root r (the most general concept) is unique. Given two concepts c' and c'', let $< r = p'_1, \ldots, p'_{n'} = c' >$ and $< r = p''_1, \ldots, p''_{n''} = c'' >$ be their paths to the root. Now, their closest common ancestor is uniquely identified, as the last element in common in the two paths, say p_k (i.e., $\forall i = 1, \ldots, k : p'_i = p''_i = p_i$). This determines three sub-paths: the sub-path in common ($< p_1, \ldots, p_k >$) and the two trailing sub-paths ($< p'_{k+1}, \ldots, p'_{n'} >$ and $< p''_{k+1}, \ldots, p''_{n''} >$). The former can be interpreted as the common information, and the latter as the residuals, and hence their lengths ($n' - k, k, n'' - k$) can serve as arguments (n_a, l_a, m_a) to apply the similarity formula [6]. This represents a novelty with respect to other approaches in the literature, where only (one or both of) the trailing parts are typically exploited.

[2] According to [2], this yields a similarity measure rather than a full semantic relatedness measure, but we are currently working to extend it by taking into account other relationships as well.

Table 1. Sample similarity values between WordNet words/concepts

Concept	**Concept**	**Similarity**
mouse (animal) [102330245]	computer (device) [103082979]	0.394
mouse (device) [103793489]	computer (device) [103082979]	0.727
mouse (device) [103793489]	cat (pet) [102121620]	0.384
mouse (animal) [102330245]	cat (pet) [102121620]	0.775
cat (domestic) [102121620]	computer (device) [103082979]	0.384
cat (pet) [102121620]	tiger (animal) [102129604]	0.849
cat (wild) [102127808]	tiger (animal) [102129604]	0.910
cat (pet) [102121620]	dog (pet) [102084071]	0.627
dog (pet) [102084071]	horse (domestic) [102374451]	0.542
horse (domestic) [102374451]	horse (chess) [103624767]	0.339
mouse (animal) [102330245]	mouse (device) [103793489]	0.394

that, according to the one-domain-per-discourse assumption, exploits the closest pair of concepts associated to those words. The similarity between groups of words (if needed) can be computed by pairwise working on the closest (i.e., taxonomically most similar) words in each group.

Let us now show the effectiveness of the proposed approach. Clearly, we need a taxonomy to be used as a background knowledge. Since manually setting up a general taxonomy is a hard work, here we will exploit the most famous taxonomy available nowadays, WordNet (WN) [11], that provides both the conceptual and the lexical level. While this helps the demonstration, however, it is important to stress the fact that the technique works on any taxonomy. Table 1 reports the similarity values corresponding to some prototypical and tricky pairs of concepts/words. At the level of concepts, the similarity ranking is quite intuitive: the closest pairs are 'wild cat'-'tiger' and 'pet cat'-'tiger', followed by 'mouse animal'-'pet cat', then by 'mouse device'-'computer device', by 'pet cat'-'dog pet' and by 'dog pet'-'horse animal', all with similarity values above 0.5. Conversely, less related concepts receive a lower value: all odd pairs, mixing animals and devices or objects (including polysemic words), get low values, below 0.4.

The proposed technique might be usefully exploited as a support to further processing such as logic deductions or Natural Language Processing. For instance, assuming consistency of domain among the words used in a same context [8], by pairwise comparing all concepts underlying two synonymic or polysemic words, it might suggest a ranking of which are the most probable senses for each, this way serving as a simple Word Sense Disambiguation [7] procedure, or as a support to a more elaborate one. Referring back to Table 1, one might disambiguate a polysemic word (e.g., 'mouse') by comparing its possible underlying concepts to the other concepts that are present in the same text (e.g., 'cat' and 'dog' rather than 'computer'). Or, one might be interested in ranking a set of candidate concepts by closeness with respect to a given concept (e.g., ranking 'dog (pet)', 'tiger (animal)' and 'cat (wild)' with respect to 'cat (pet)'), etc.

4 Numeric Similarity

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 traditionally tackled in two different ways. Plugging the ability to handle numeric information directly in the inference engine somehow 'spoils' its behavior and adds complexity (reducing efficiency). Another solution aimed at keeping the purely logical setting, to preserve applicability of the logical representation and inference techniques, consisted in a discretization of the range of numeric values allowed for a given attribute into pre-defined intervals, associated to corresponding symbolic descriptors (e.g., size_small, size_large, ...; position_top, position_middle, ...).

The latter option requires a pre-processing of the original descriptions to turn all instances of numeric attributes into the corresponding discretized descriptors. But, what is the correct 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. Although techniques for (semi-)automatic definition of the intervals have been proposed (e.g., [1]), based on statistics on the occurrence and distribution of values for the attribute, a manual intervention is often required to constrain and/or fix their outcome. In any case, if the intervals are not to be considered as completely distinct entities, 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).

A further, general problem is 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 285 meters might be meaningful when comparing two fields, but completely insignificant when comparing planets according to their size). All these considerations show that, in an extended framework considering interpreted predicates and constants in addition to simple syntactic entities, the ability to handle numeric information is fundamental.

Table 2. 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

Numeric information can take the form of a single value (e.g., a specific measurement in an observation) or of an interval (e.g., a range of allowed values in a model). Thus, the comparison technique must apply to all possible combinations thereof: two intervals, an interval and a value, two values. Let us start our discussion from the case of two intervals, say $I' = [i'_1, i'_2]$ and $I'' = [i''_1, i''_2]$. Two intuitive approaches are available to extract parameters l, n and m for numeric comparisons (assume, without loss of generality, that $i'_1 \leq i''_2$):

1. considering the distance between the interval **extremes**: $n = |i''_1 - i'_1|$, $m = |i''_2 - i'_2|$ and $l = \min(i'_2, i''_2) - \max(i'_1, i''_1)$ if non-negative (or 0 otherwise)[3].
2. considering the intervals as **sets**, and exploiting set operators: $l = ||I' \cap I''||$ would be the width (expressed as $|| \cdot ||$) of the overlapping part, and $n = ||I' \setminus I''||, m = ||I'' \setminus I'||$ their symmetric differences, respectively.

The behavior of the two candidate approaches on a set of sample intervals is shown in Table 2. 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. 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 intuitive (the farther two intervals, the more different they are), on the other hand, in the case of an interval being a sub-interval of the other it is not. Indeed, the set-based approach charges the whole 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.

Both strategies can be straightforwardly applied also to the comparison of an interval to a single value, considered as an interval in which the two extremes coincide. In this case one gets always $l = 0$, and partial overlapping never happens. Thus, the features of the two approaches become more evident, as shown in

[3] 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.

Table 3. 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	$0.\overline{05}$	$0.\overline{3}$	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

Table 3. 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. Given a value included in an interval, their similarity according to the set-based approach is constant (it depends only on the width of the interval, not on the specific single value), 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 the similarity; as long as the value approaches the interval extremes, the similarity grows up to the same similarity as the set-based approach. This is counterintuitive, because if the interval specifies an allowed range, either there is no reason to prefer particular regions within that range, or one would rather prefer a value in the middle than one near the extremes. 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 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, 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, that 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 height of a

Table 4. 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	$0.\overline{3}$

building). We propose the following formula:

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

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 general case, 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. Some sample comparisons between single values are reported in Table 4 (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, while (1) can be used when at least an interval is involved. The set-based strategy is better in the case of an interval or value being included in another interval, because it better fits the spirit of the similarity function parameters. Conversely, in the case of disjoint intervals the extreme-based strategy is able to consider the actual distance from the interval and/or value extremes, which affects the residual parameters. Both strategies are equivalent in the case of partially overlapping intervals. Overall, a cooperation of the three strategies is desirable: a deeper empirical study is planned as future work, to establish if and how a smooth combination thereof can be obtained, ensuring comparable similarity assessments among the 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).

5 Extension of the First-Order Logic Framework

We can now discuss where to embed the new similarity perspectives in the overall First-Order Logic similarity framework. Since taxonomic and numeric predicates represent further information about the objects involved in a description, in addition to their properties and roles, term similarity is the proper component to be extended:

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

where the components can be weighted differently if needed, and the additional components express the taxonomic and numeric similarity associated to the two terms, as specified above. Of course, we assume that there is some way to distinguish taxonomic and numeric predicates from ordinary ones, so that they can be specifically handled by the procedures.

5.1 Discussion

To the best of our knowledge, there is no other attempt in the literature to mix in a single similarity framework all these different kinds of information (relational, taxonomic, and numeric). As to each kind taken separately:

- There are a few proposals to assess similarity between first-order logic descriptions, which were compared in [5] to the technique for first-order Horn clauses adopted as a basic framework in this paper.
- Several techniques are available for taxonomic similarity, usually working on WordNet (a selection of which is presented in [2]), but they generally adopt local approaches based on the length of specific paths connecting the two items under comparison, while our global approach takes into account all the available information in the taxonomy related to them.
- Numeric similarity between points in a multi-dimensional space is usually computed using geometric approaches such as Euclidean distance or Cosine similarity, whereas our technique separately works on each dimension and is able to deal also with intervals (that are likely to be found in a general model, while specific values are typical of observations).

Motivations for developing the taxonomic and numeric similarity techniques proposed in this paper came both from the above mentioned advantages over previous proposals, and from the need to adopt the same similarity formula as the general framework for the sake of consistency. Differently from the taxonomic case, the solution for numeric information required different approaches, depending on the overlapping situation between intervals, and a specific formula for the case of single values, to suitably capture the peculiarities of each possible case.

In the proposed solution, the structural information conveyed by the relationships in the first-order logic descriptions is exploited to constrain possible terms associations, and then the similarity between terms is used to fine-tune such associations and determine the overall similarity between those descriptions.

Specifically, taxonomic and numeric information takes the form of additional attributes of the terms (representing objects) in the first-order logic descriptions under comparison, and hence contributes (along with their properties and roles as expressed by traditional logic predicates) to assess their similarity.

5.2 Sample Application of Taxonomic Extension

As an example involving taxonomic similarity, consider Natural Language Processing. Although much more computationally demanding than traditional simple bag-of-word approaches, techniques that take into account the syntactic structure of sentences are fundamental to fully capture the information they convey. Reporters know very well that, swapping subject and object in a sentence like "The dog bit the man", dramatically changes the appeal of the underlying news. The syntactic structure and relationships among discourse components in sentences can be represented by atoms in FOL Horn clauses, but it is not sufficient. For instance, the following sentences:

1. "The boy wants a small dog"
2. "The girl desires a yellow canary"
3. "The hammer hits a small nail"

structurally share the same grammatical pattern, thus no hint is available to assess which is more similar to which. Even worse, sentence 1 would appear to be closer to sentence 3 than to sentence 2, due to the presence of word 'small', while it is clear that 1 and 2 are conceptually the most similar to each other as soon as one considers that 'boy' and 'girl' are two young persons, 'to want' and 'to desire' are synonyms and 'dog' and 'canary' are two pets. But taxonomic information is needed to know this.

For demonstration purposes, let us consider a simplified structural description language for natural language sentences:

subj(X,Y) : Y is the subject of sentence X
pred(X,Y) : Y is the predicate of sentence X
dir_obj(X,Y) : Y is the direct object of sentence X
ind_obj(X,Y) : Y is the in direct object of sentence X
noun(X,Y) : Y is a noun appearing in component X of the sentence
verb(X,Y) : Y is a verb appearing in component X of the sentence
adj(X,Y) : Y is an adjective appearing in component X of the sentence
adv(X,Y) : Y is an adverb appearing in component X of the sentence
prep(X,Y) : Y is a preposition appearing in component X of the sentence

Additionally, each noun and verb in the sentence is described by a taxonomic predicate expressing the corresponding concept (or word), while other properties (including adjectives and adverbs) are expressed by ordinary unary predicates[4]. For the three sentences reported above one gets:

[4] WordNet defines heterarchies for noun and verbs, but adopts different representation structures for adjectives and adverbs. We are currently working to extend the proposed similarity strategy to handle those structures, as well.

```
s1 = sentence(s1) :- subj(s1,ss1), pred(s1,ps1), dir_obj(s1,ds1),
    noun(ss1,nss1), boy(nss1), verb(ps1,vps1), want(vps1),
    adj(ds1,ads1), small(ads1), noun(ds1,nds1), dog(nds1).
s2 = sentence(s2) :- subj(s2,ss2), pred(s2,ps2), dir_obj(s2,ds2),
    noun(ss2,nss2), girl(nss2), verb(ps2,vps2), desire(vps2),
    adj(ds2,ads2), yellow(ads2), noun(ds2,nds2), canary(nds2).
s3 = sentence(s3) :- subj(s3,ss3), pred(s3,ps3), dir_obj(s3,ds3),
    noun(ss3,nss3), hammer(nss3), verb(ps3,vps3), hit(vps3),
    adj(ds3,ads3), small(ads3), noun(ds3,nds3), nail(nds3).
```

where taxonomic predicates are shown in italics.

Syntactically, the generalization between $s2$ and both $s1$ and $s3$ is:

```
sentence(X) :- subj(X,Y), noun(Y,Y1), pred(X,W), verb(W,W1),
               dir_obj(X,Z), adj(Z,Z1), noun(Z,Z2).
```

while the generalization between $s1$ and $s3$ is:

```
sentence(X) :- subj(X,Y), noun(Y,Y1), pred(X,W), verb(W,W1),
               dir_obj(X,Z), adj(Z,Z1), small(Z1), noun(Z,Z2).
```

so that the latter pair, having in the generalization an additional literal with respect to the former pairs, would get a larger similarity value, in spite of the very different content. Using WordNet, the similarity between words is:

boy-girl = 0.75	boy-hammer = 0.436	girl-hammer = 0.436
want-desire = 0.826	want-hit = 0.361	desire-hit = 0.375
yellow-small = 0.563	small-small = 1	
dog-canary = 0.668	dog-nail = 0.75	canary-nail = 0.387

which, embedded in the overall clause similarity computation, overcomes the wrong similarity assessment according to syntactic comparisons only, and fixes it neatly assigning to the first two sentences the largest similarity value (also in spite of the odd 'dog'-'nail' similarity)[5]:

fs(s1,s2) = 1.770   fs(s1,s3) = 1.739   fs(s2,s3) = 1.683

5.3 Sample Application of Numeric Extension

As regards numeric information, the values/intervals appearing as predicate arguments are by themselves sufficient to identify it. 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 are represented by binary predicates $a(t, v)$ meaning that "object t has (numeric) value v for attribute a". The case of relationships associated with numeric information (e.g., the weight of arcs in a graph), can be 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 $arc(n_1, n_2, a), weight(a, v)$).

[5] All similarities agree with intuition, except the pair dog-nail that gets a higher similarity value than dog-canary. This is due to the interpretations of 'dog' as "a hinged catch that fits into a notch of a ratchet to move a wheel forward or prevent it from moving backward" and 'nail' as "a thin pointed piece of metal that is hammered into materials as a fastener".

A full evaluation of the numeric similarity strategy would require a dataset involving numeric attributes and including both intervals and specific values. Unfortunately, the available datasets usually fill each numeric attribute with a single number expressing its value for the object being described, while intervals are typically 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. A good testbed for assessing numeric similarity performance is the supervised clustering task: after hiding the class specification from the examples, we provide the resulting observations to a standard k-means algorithm and ask it to split the whole dataset into different groups based on the similarity measure described in previous sections. Then, the performance of the resulting clusters is checked according to the *purity* measure (i.e., the percentage of correct association in the best overlapping between the clusters and the actual classes that were hidden), that can be considered as the counterpart of accuracy in supervised learning.

Our experiment concerns the problem of distinguishing documents according to their layout description, using a dataset 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 predicates (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 of the block in the page, and width/height thereof. The expectation is that these descriptors are very significant to class discrimination. Previous applications on this dataset [4] 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 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. The k-means algorithm was 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 among all pairs of observations ($353 \cdot 352/2 = 124256/2 = 62128$ comparisons), the resulting clusters are shown in Table 5.

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

Table 5. Dispersion matrix for the Document Clustering problem

	Elsevier	MLJ	JMLR	SVLN	Total	Errors	Purity
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	–	–	–

of the various classes/clusters. Compared to the 92.35% purity reported in [5] 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). 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 MLJ 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.

Results of additional clustering tasks, each pairwise involving two of the four classes in the dataset, show purity values ranging between 81.25% (for the MLJ-SVLN pair) and 98.82% (for the JMLR-SVLN pair), with an average of 89.99%.

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 taxonomic (concepts or words) or numeric (intervals and/or single values) background knowledge in order to capture unexpressed and underlying relationships among the concepts described.

This paper defined specific strategies for assessing similarity between these two kinds of information, discussed their integration as an extension of a general framework for assessing the similarity between Horn clauses, and reported experimental results that show the effectiveness of the proposal. Experiments on the taxonomic component concerned Natural Language Processing using a state-of-the-art taxonomy as a background knowledge, while experiments on the numeric component concerned a clustering task on a typical dataset mixing relational and numeric information.

As to the taxonomic component, future work will concern fine-tuning of the similarity computation methodology by exploiting additional taxonomic relationships (other than hypernymy). As to the numeric component, future work will concern deeper empirical evaluation of the proposed approaches in the case of intervals, and the smooth cooperation of the different sub-strategies defined in this paper. As to the integration of the new components in the general similarity framework, we plan to study the relative importance of the single components for term similarity evaluation (properties, roles, taxonomy, numeric), to determine suitable weighting schemes for them. Then, experiments on the 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] M. Biba, F. Esposito, S. Ferilli, N. Di Mauro, and T.M.A. Basile. Unsupervised discretization using kernel density estimation. In *IJCAI'07*, pages 696–701, 2007.

[2] A. Budanitsky and G. Hirst. Semantic distance in wordnet: An experimental, application-oriented evaluation of five measures. In *Proc. Workshop on WordNet and Other Lexical Resources, 2nd meeting of the North American Chapter of the Association for Computational Linguistics.* Pittsburgh, 2001.

[3] S. Ceri, G. Gottlöb, and L. Tanca. *Logic Programming and Databases.* Springer-Verlag, Heidelberg, Germany, 1990.

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

[5] S. Ferilli, T. M.A. Basile, M. Biba, N. Di Mauro, and F. Esposito. A general similarity framework for horn clause logic. *Fundamenta Informaticæ*, 90(1-2):43–46, 2009.

[6] S. Ferilli, M. Biba, N. Mauro, T. M. Basile, and F. Esposito. Plugging taxonomic similarity in first-order logic horn clauses comparison. In *Emergent Perspectives in Artificial Intelligence*, Lecture Notes in Artificial Intelligence, pages 131–140. Springer, 2009.

[7] Nancy Ide and Jean Vronis. Word sense disambiguation: The state of the art. *Computational Linguistics*, 24:1–40, 1998.

[8] Robert Krovetz. More than one sense per discourse. In *NEC Princeton NJ Labs., Research Memorandum*, 1998.

[9] 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.

[10] J. W. Lloyd. *Foundations of Logic Programming.* Springer-Verlag, Berlin, second edition, 1987.

[11] George A. Miller. Wordnet: A lexical database for English. *Communications of the ACM*, 38(11):39–41, 1995.

[12] S. Muggleton. Inductive logic programming. *New Generation Computing*, 8(4):295–318, 1991.