A New Similarity Measure for Guiding Generalizations Search

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

1 Introduction

Few works are available in the literature to define similarity criteria between First-Order Logic (FOL) formulæ, where the presence of relations causes various portions of one description to be possibly mapped in different ways onto another description, which poses serious computational problems. Hence, the need for a set of general criteria that are able to support the comparison between formulæ. This could have many applications: making a subsumption procedure converge quickly towards the correct associations, developing a flexible matching procedure, supporting Case-based reasoning and k-Nearest Neighbor techniques, grouping observations into homogeneous concepts (*conceptual clustering*), helping theory revision systems to choose the best definition to be refined. In this paper we tackle the case of two descriptions (e.g., a definition and an observation) to be generalized, where the similarity criteria could help in focussing on the subparts of the descriptions that are more similar and hence more likely to correspond to each other, based only on their syntactic structure. In particular, we focus on FOL formulæ in the form of clauses, of interest to ILP, and specifically to the case of linked Datalog clauses, without loss of generality [2].

2 Similarity Criteria and Formula

Intuitively, a similarity criterion between two items might be based both on the number of common features l, which should concur positively, and also on the numbers n and m of features of each description that are not owned by the other, which should concur negatively. We developed the following novel similarity formula, where α weights the importance of either item:

$$\mathrm{sf}(\alpha, n, l, m) = \alpha \frac{l+1}{l+n+2} + (1-\alpha)\frac{l+1}{l+m+2} \tag{1}$$

In a clause, terms represent specific objects, whose properties and features are generally expressed by unary predicates, and whose relationships are expressed by n-ary predicates. Accordingly, two levels of similarity can be defined for pairs of first-order descriptions: the *object* level and the *structure* one.

As to object similarity, given two distinct objects (terms) a' and a'', two kinds of features can be distinguished: the properties they own (*characteristic features*), expressed by unary predicates, and the ways in which they relate to other objects (*relational features*), expressed by the position the object holds among an n-ary predicate arguments, corresponding to different roles played by the objects. Two corresponding similarity values can be associated to a' and a'': a *characteristic similarity*, where (1) is applied to n, l and m computed on the characteristic features, and a *relational similarity*, based on how many times the two objects play the same or different roles in the n-ary predicates. Here, the arguments for (1) are given by the sum over all possible roles played by any of the two objects of the n_R, l_R and m_R values for each role R.

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. Given an n-ary literal, we define its *star* as the multiset of predicates corresponding to the literals linked to it by some common argument. Thus, any two compatible n-ary literals l' nd l'' can be compared by applying (1) to the number of common and different predicate items in each of the two stars, and adding all characteristic and relational similarities for each pair of their arguments in corresponding positions. Each clause can be represented as a Directed Acyclic Graph (in which literals are the nodes) *stratified* such that the head is the only node at level 0, and each successive level introduces nodes not yet reached by edges by setting an incoming edge to them from each node in the previous level having among its arguments at least one term in common with it. Now, all possible paths from the head to leaf nodes (those with no out-coming edges) can be interpreted as the basic components of the overall structure of the clause, and the *structural similarity* between any two paths p' and p'' taken from the two clauses can be computed by applying (1) to the length l of their compatible (as to predicates and arguments bindings) initial sequence and the values n and m of the remaining sequences in each, plus the star similarity of all couples of literals in the initial sequence.

Finally, the path pairs can be ordered by decreasing similarity, and a generalization can be computed by starting from the top and going down the ranking, adding to the partial generalization generated thus far the common literals of each pair whenever they are compatible. Further generalizations can then be obtained through backtracking.

3 Experiments

The similarity-driven generalization procedure, using (1) with $\alpha = 0.5$, was compared to a previous non-guided procedure, embedded in the learning system INTHELEX [1], on a dataset of 122 descriptions representing scientific paper first pages layout, belonging to 4 different classes. This gave rise to 488 positive/negative examples for the classification task, and to 1488 examples for 12

Table 1. Experimental results

		Ratio	Time (sec.)	Cl	Gen	Exc^+	$Spec^+$	$Spec^-$	Exc^-	Acc
Classification	SF	90.52	579	8	47(+19)	0	2	0	0	0.94
	I	70.22	137	7	33(+66)	0	1	1	1	0.97
	S80	73.63	206	7	33(+19)	0	0	1	1	0.97
Component Labelling	SF	91.09	22220	36	180(+201)	0	8	3	3	0.89
	I	68.85	33060	39	137(+2500)	0	15	11	12	0.93
	S80	71.75	15941	54	172(+660)	0	14	8	2	0.93

concepts for the significant component labelling task. 10-fold cross-validation was exploited to assess predictive accuracy.

In 47 correct generalizations, the similarity-driven generalization (SF in Table 1) preserved on average 89,71% literals of the shortest clause, with a maximum of 99,24% (131 literals out of 132, against an example of 333) and just 0,01 variance. Hence, the produced generalization is likely to be very near to the least general one, as confirmed by the fact that when the first generalization produced was not consistent with all past negative examples (which happened in 20 cases in which the starting clause was already extremely general) no more specific generalization was found within the next 500 attempts (a limit set to avoid the system to deadlock on some generalizations). Noteworthy, the non-guided generalization procedure with unbound search (I in Table 1) was never able to find correct generalizations within the first 500 attempts. However, such specific generalizations show low predictive accuracy with respect to the INTHELEX algorithm, probably due to the need of more examples in order to converge to more predictive definitions or to overfitting, both being a consequence of less general generalizations. For this reason, a threshold was set on the similarity-driven generalization (S80 in Table 1), so that it should discard 20% literals of the shortest original clause. Such a modified version took less than 1/3 runtime to complete the learning task with respect to the unbound version; the number of generalizations significantly reduces of 1/5, but at the cost of negative literals and exceptions as specializations. However, the behavior strictly resembles that of the old generalization, but with nearly 2/3 runtime savings. The difference of 5 additional generalizations is probably due to the use of similarity yielding generalizations more tight to the examples: indeed, as in the unbound case, no more specific generalization is ever found within the first 500 attempts, whereas the INTHELEX procedure also computed and tried 66 inconsistent generalizations in 5 additional unsuccessful cases.

References

[1] F. Esposito, S. Ferilli, N. Fanizzi, T. Basile, and N. Di Mauro. Incremental multi-strategy learning for document processing. *Applied Artificial Intelligence Journal*, 17(8/9):859–883, 2003.

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