A Language-Based Similarity Measure

Lionel Martin and Frédéric Moal

LIFO - Université d'Orléans,
rue Léonard de Vinci, BP 6759,
45067 Orleans cedex 2, FRANCE
{martin,moal}@lifo.univ-orleans.fr

Abstract. This paper presents an unified framework for the definition of similarity measures for various formalisms (attribute-value, first order logic...). The underlying idea is that the similarity between two objects does not depend only on the attribute values of the objects, but more especially on the set of the potentially relevant definitions of concepts for the problem considered.
In our framework, the user defines a language with a grammar to specify the similarity measure. Each term of the language represents a property of the objects. The similarity between two objects is the probability that these two objects both satisfy or both reject simultaneously the properties of the given language. When this probability is not computable, we use a stochastic generation procedure to approximate it.
This measure can be applied for both clustering and classification tasks. The empirical evaluation on common classification problems shows a very good accuracy.

1 Introduction

Distance or similarity measures play a central role in many machine learning problems. Usually, the definition of similarity between objects depends on the formalism used for the object description. Two main approaches have been proposed: weight-based similarity measure, relying on a weighted sum of similarity of different parts of objects (used in attribute-value languages) and description-based similarity measure based on the construction of object descriptions, used in both attribute-value languages and first order logic.

In the case of attribute-value languages, objects are vectors and the similarity between objects depends on the similarity of vectors components; two types of attributes are separately considered: nominal attributes (having value in an unordered finite -and generally small- set of symbolic values) and numerical ones. In most cases, two different values of a nominal attribute are not similar (similarity=0), otherwise they are similar (similarity=1): in this case, the similarity between two objects is the number of common attributes. In some cases, the frequency of values is used to define the similarity of nominal attributes [CS93, SW86]. Numerical attributes can be considered in various ways: the similarity can be defined from their Euclidean distance or their normalized distance [Aha89,GCM95] or values are discretized and then considered as nominal attributes. The final similarity between tuples is the sum of the similarity induced

L. De Raedt and P. Flach (Eds.): ECML 2001, LNAI 2167, pp. 336–347, 2001.

by the nominal attributes and the similarity induced by the numerical attributes. This sum can be weighted with different techniques: weights are computed by statistical techniques [MT94], based on conditional probabilities [SW86], or are obtained by an optimization technique such as a genetic algorithm.

Another way to build a similarity function consists of producing a description of objects. When such descriptions are sets (of atoms), the similarity is based on description intersections [Bis92,EW96]; when descriptions are rules, the similarity between two objects is given by the number of rules satisfied by the two objects [Seb97,SS94]. RISE [Dom95] builds rules for attribute-value objects and proposes an extension of weight-based similarity in the case of generalized attribute-value vectors.

In any case, two objects are considered to be similar when they share some properties. However, it is important to notice that the main expectation from a similarity measure is the ordering induced, i.e. such measure is used to state that object e_1 is more similar to e_2 than e_3 is. Then e_1 can be considered to be more similar to e_2 than e_3 is, if e_1 shares more properties with e_2 than with e_3. This is achieved by several rule-based similarity measures [Dom95,SS94,Seb97,EW96] which consider a finite set of properties, made of rules learned to characterize some objects. The common point of these approaches is that the considered rules have a discriminating power: a rule is built to discriminate a set of examples from a set of counter-examples, and then the similarity is induced in a supervised manner. So this set of rules is strongly biased by the original set of objects and the induction technique used.

In this paper, we propose to base the similarity measure on a language, representing a possibly infinite set of properties or rules. This language is composed of terms that represent concept descriptions. Each term of the language is associated with an evaluation function, able to test whether an object satisfies the property or not. In the case of a finite language, the similarity between two objects is given by the number of properties both objects satisfy or reject simultaneously. In the case of infinite languages, we propose to approximate the similarity by randomly generating a finite subset of the language.

The next section formalizes our similarity definition for finite languages. We focus on the properties of this similarity and its limitations for infinite languages. The Section 3 presents a constraint tree grammar formalism used to specify a language and to generate a finite subset of this language. In Section 4, we present experimental results of this similarity: it has been implemented for a classification task based on a nearest neighbor algorithm and tested on some well-known attribute-value problems.

2 A Similarity Measure

We consider a set of objects $\mathcal{E} = \{e_1, \ldots, e_m\}$ and a language $\mathcal{L}$, i.e. a (possibly infinite) set of terms $\{t_1, \ldots, t_n, \ldots\}$. In this paper, a term $t \in \mathcal{L}$ represents a property (for example *color=blue*); for each object $e \in \mathcal{E}$, either e satisfies t (noted $t(e) = 1$) or e does not satisfy t (noted $t(e) = 0$). We will consider further (Section 2.5) the case where, due to incomplete or imprecise data, we cannot state that $t(e) = 1$ or $t(e) = 0$.

Now, given two objects e_i and e_j in $\mathcal{E}$ and a property $t \in \mathcal{L}$, we say that e_i and e_j are equivalent w.r.t. t if $t(e_i) = t(e_j)$ (e_i and e_j can not be distinguished by t). When e_i and e_j are equivalent w.r.t. most terms (resp. a few terms) of $\mathcal{L}$, we propose to consider that the similarity between e_i and e_j be high (resp. low). We propose to formalize this similarity by using a probability measure.

2.1 Finite Languages

Let $\mathcal{L}$ be a language. A probability measure [DeG86] is defined over a space $(\Omega, \mathcal{B})$ where $\mathcal{B}$ is an α-algebra over the basic space Ω. We define the basic space $\Omega = \mathcal{L}$, i.e. terms $t \in \mathcal{L}$ are elementary events.

In the case of finite languages $\mathcal{L}$, the space $(\mathcal{L}, \mathcal{P}(\mathcal{L}))$, where $\mathcal{P}(\mathcal{L})$ is the powerset of $\mathcal{L}$, is probabilisable. The similarity between e_i and e_j can then be defined as the probability that e_i and e_j are equivalent w.r.t. a term t randomly chosen in $\mathcal{L}$. Let $\mathcal{L}_{e_i,e_j} = \{t \in \mathcal{L} | t(e_i) = t(e_j)\}$ be the set of terms of $\mathcal{L}$ for which e_i and e_j are equivalent; $\mathcal{L}_{e_i,e_j} \in \mathcal{P}(\mathcal{L})$ and we define the previous similarity as the probability of the event $\mathcal{L}_{e_i,e_j}$

Definition 1. *Let $\mathcal{L}$ be a finite language and P a probability measure over $(\mathcal{L}, \mathcal{P}(\mathcal{L}))$. Let e_i and e_j be two objects of $\mathcal{E}$ and $\mathcal{L}_{e_i,e_j} = \{t \in \mathcal{L} | t(e_i) = t(e_j)\} \in \mathcal{P}(\mathcal{L})$. The similarity with respect to $\mathcal{L}$ between e_i and e_j, noted $sim_{\mathcal{L}}(e_i, e_j)$, is defined by $sim_{\mathcal{L}}(e_i, e_j) = P(\mathcal{L}_{e_i,e_j})$.*

In the following, we will write $sim(e_i, e_j)$ for $sim_{\mathcal{L}}(e_i, e_j)$ when $\mathcal{L}$ is not ambiguous.

In the case of a uniform probability over $\mathcal{L}$, i.e. each term $t \in \mathcal{L}$ has the same probability: $P(t) = \frac{1}{|\mathcal{L}|}$, this definition is equivalent to

$$sim(e_i, e_j) = \frac{1}{|\mathcal{L}|} . \sum_{t \in \mathcal{L}} \delta_t(e_i, e_j)$$

where $\delta_t(e_i, e_j) = 1$ if $t(e_i) = t(e_j)$ and $\delta_t(e_i, e_j) = 0$ if $t(e_i) \neq t(e_j)$.

In this definition, the similarity between e_i and e_j is the rate of terms $t \in \mathcal{L}$ such that $t(e_1) = t(e_j)$. Previous works propose a similarity measure based on a sum of weights associated with learned rules [SS94], attributes [CS93,SW86] or occurrences [Bis92]. In the previous definition, the sum is not weighted and is computed for any term of the language.

2.2 Example

In this example, objects are glasses of colored water. Each object is described by a couple $< col, temp >$ where *col* (resp. *temp*) is the color (resp. temperature expressed in fahrenheit degrees) of the object.

$e_1 =< green, 14 >$	$e_3 =< orange, 221 >$
$e_2 =< red, 203 >$	$e_4 =< pink, 266 >$

Language $\mathcal{L}_1$. The first language considered here focuses on the temperature of the water; it contains 30 terms: $\mathcal{L}_1 = \{temp > 0, temp > 10, \ldots, temp > 290\}$ and we consider a uniform probability over $\mathcal{L}_1$. For each e_i and e_j, we can compute $sim(e_i, e_j)$: for example, $sim(e_1, e_2) = 11/30$, $sim(e_1, e_3) = 9/30$, ... and we can see that the most similar objects w.r.t $\mathcal{L}_1$ are e_2 and e_3 which is intuitive if we focus on the temperature.

Language $\mathcal{L}_2$. The second language considered here focuses on the chemical state of the water (gaseous, liquid, solid): the language $\mathcal{L}_2$ is defined by two terms: $\{temp \leq 32, temp \leq 212\}$ and we consider a uniform probability over $\mathcal{L}_2$. With respect to this language, the most similar objects are e_3 and e_4 ($sim(e_3, e_4) = 1$). Moreover, $sim(e_2, e_4) = 1/2$ and $sim(e_1, e_4) = 0$, i.e the object e_4 (gaseous) is less far from the object e_2 (liquid) than the object e_1 (solid).

Let us notice that if we had added an attribute *state* to the object description, the language $\mathcal{L}'_2 = \{state = solid, state = liquid, state = gazeous\}$ would lead to a different result since then, $sim(e_3, e_4) = 1$ but $sim(e_2, e_4) = 1/3$.

Language $\mathcal{L}_3$. Let us consider now a language focusing on the color of objects. With respect to the language $\{col = green, col = pink, col = red, col = orange\}$, for any e_i, e_j with $e_i \neq e_j$, we have $sim(e_i, e_j) = 1/2$. In such cases, we would like to use a distance over the domain of color, indicating that red and pink are more similar than red and orange, ... In this case, we propose to extend the previous language by adding {*(color=red or color=pink), (color=red or color=orange),* ...}. With this language, e_2 and e_4 share the property *(color=red or color=pink)* and then $sim(e_2, e_4) > sim(e_1, e_4)$. Finally we can consider a non-uniform probability over $\mathcal{L}_3$, given a greater weight to *(color=red or color=pink)* than to *(color=red or color=orange)*. Finally, we can consider the following language (the probability of each term is written in boxes) : $\mathcal{L}_3$ = {*col=green* $\boxed{20\%}$, *col=pink* $\boxed{20\%}$, *col=red* $\boxed{20\%}$, *col=orange* $\boxed{20\%}$, *(color=red or color=pink)* $\boxed{12\%}$, *(color=red or color=orange)* $\boxed{6\%}$, *(color=pink or color=orange)* $\boxed{2\%}$ }. With such a language, the most similar objects are now e_2 and e_4 ($sim(e_2, e_4) = 20 + 0 + 0 + 20 + 12 + 0 + 0 = 52\%$).

Preliminary conclusion. In this example, the similarity between objects depends on the language considered. It shows that the similarity is not intrinsic to the objects, but depends on the point of view by which objects are considered, i.e. on properties (or concept definitions) that are relevant to the user. The language can then be viewed as a central bias on the similarity definition. We propose in Section 3 to use a grammar to express such a bias.

2.3 A Pseudo-Distance

For any $e_i, e_j \in \mathcal{E}$, $sim(e_i, e_j) \in [0, 1]$; a dissimilarity measure can be naturally defined as $dissim(e_i, e_j) = 1 - sim(e_i, e_j) = P(\overline{\mathcal{L}_{e_i,e_j}})$. It is easy to see that *dissim* is symmetric. To show that *dissim* satisfies the triangle inequality, i.e. for any e_1, e_2, e_3, $dissim(e_1, e_3) \leq dissim(e_1, e_2) + dissim(e_2, e_3)$, we have to

prove that $P(\overline{\mathcal{L}_{e_1,e_3}}) \leq P(\overline{\mathcal{L}_{e_1,e_2}}) + P(\overline{\mathcal{L}_{e_2,e_3}})$. For any $t \in \mathcal{L}$, if $t(e_1) \neq t(e_3)$ then either $t(e_1) \neq t(e_2)$ or $t(e_2) \neq t(e_3)$ which prove the triangle inequality.

Since $dissim(e, e) = 0$ for any e, $dissim$ is a pseudo distance. However $dissim$ is not a distance since it may exist $e_i, e_j \in \mathcal{E}$ with $dissim(e_i, e_j) = 0$ but $e_i \neq e_j$. This case happens when the language does not allow to distinguish two objects $(\forall t \in \mathcal{L}, t(e_i) = t(e_j))$: in Example 2.2, objects $e_5 =< blue, 18 >$ and $e_1 =< green, 14 >$ are equivalent w.r.t. $\mathcal{L}_1$ ($dissim_{\mathcal{L}_1}(e_1, e_5) = 0$); for practical purpose, e_1 and e_2 are equivalent but they are not equal.

2.4 Usual Languages

For objects described by a set of attributes $A = \{Att_i\}$, with a domain D_i for Att_i with nominal values, a (finite) language can be defined by $\mathcal{L}_{A=v} = \{Att_i = v_{i_j} | v_{i_j} \in D_i\}$.

The dissimilarity associated with this language is a distance equivalent to the Hamming distance if each attribute is binary.

If we consider objects described by a set of numerical attributes $A = \{Att_i\}$, various languages can be considered:

$\mathcal{L}_{A=n} = \{Att_i = v_{i_j}\}$
$\mathcal{L}_{A\leq x} = \{Att_i \leq v_{i_j}\}$
$\mathcal{L}_{A\geq x} = \{Att_i \geq v_{i_j}\}$
$\mathcal{L}_{x\leq A\leq y} = \{v_{i_k} \leq Att_i \leq v_{i_j}\}$
...

where v_{i_j} is chosen among the possible values of Att_i or is taken from a given range.

We can also consider languages such as $\{Exp \leq v\}$ where Exp is a numeric expression (linear or not) involving attributes, such as

$\mathcal{L}_{Lin} = \{a_0^i + a_1^i Att_1 + \ldots + a_n^i Att_n \geq 0 | a_k^i \in I\!R\}$

Finally, any combination of these languages can be considered, also extended with conjunctions and disjunctions.

For object described by atoms built with a predicate p, the language can be defined as the set of clauses (constrained or not) having the predicate p for head:

$\mathcal{L}_{FOL}(p) = \{h \leftarrow l_1, \ldots l_n | h = p(X_1, \ldots, X_k)\}$.

For example, consider objects called $a, b, c, \ldots$, where some objects are contained into others. This can be represented in a first order theory with predicates *contains*, *shape* and *nb_elements*: an example is shown in Fig. 1.

With this representation, we can define a similarity measure between atoms $object(a), object(b), \ldots$ with respect to the set of clauses:

$object(X) \leftarrow shape(X, circle)$.
$object(X) \leftarrow shape(X, square)$.
$object(X) \leftarrow shape(X, triangle)$.
$object(X) \leftarrow contains(X, Y), shape(Y, circle)$.
$object(X) \leftarrow contains(X, Y), shape(Y, square)$.
$object(X) \leftarrow contains(X, Y), shape(Y, triangle)$.

Two objects are similar if they are both covered or rejected by the same clauses. In the previous example, the similarity between $object(b)$ and $object(f)$ is maximum ($sim(object(b), object(f)) = 1$).

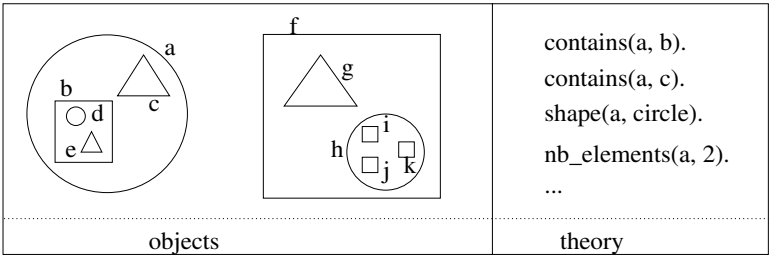

Fig. 1. Example in first order logic

We can also consider a language involving the number of objects (predicate *nb_elements*) by considering clauses such as

$object(X) \leftarrow nb\_elements(X, N), N \geq 2.$

In this framework, the similarity measure is different from [HWB01] and [Pla95] which is based on the structure of terms.

In the case of infinite languages, it can be more complicated or impossible to build a probability measure. Moreover, even for finite languages, it can be hard to compute exactly the similarity between two objects. For these reasons, we propose to define the similarity using a finite subset of the language, stochastically generated. In the next section, we propose a grammatical formalism to generate such sub-language.

2.5 Handling Missing and Constrained Values

The previous similarity definition can be extended to treat missing values: when a value is missing in e_i, it may be impossible to evaluate $t(e_i)$ for some $t \in \mathcal{L}$. Let $\mathcal{L}_{(e_i)}$ be the set of $t \in \mathcal{L}$ such that $t(e_i)$ can be evaluated. To express the similarity between e_i and e_j, the language is restricted to terms t such that both $t(e_i)$ and $t(e_j)$ can be evaluated:

$sim_{\mathcal{L}}(e_i, e_j) = sim_{\mathcal{L}_{e_i} \cap \mathcal{L}_{e_j}}(e_i, e_j)$

Moreover, attribute-value languages do not allow to handle approximate values. Consider an object for which an attribute is not precisely known, but is known to be ≤ 10. Such an attribute is then either considered unknown (value ?) or is specified by a symbolic attribute, which require that each value for this attribute is symbolic.

The similarity proposed here may support an attribute-value specification, with constraints: an object is then a constrained vector $c\Box < x_1, \ldots, x_n >$ where c express constraints on variables x_i. For example,

$e_6 \ : \ (temp \leq 10) \ \Box \ < col, temp >$

is a constrained vector for which the color is unknown and temperature is not precisely known, but known to be $\leq 10°F$.

With this formalism, a term $t \in \mathcal{L}$ can be expressed using constraints (like all languages of example 2.2) and we note $\neg t$ the negation of t; in most cases (languages made with equalities, inequalities, disequalities, conjunction, disjunction), $\neg t$ is also a constraint. Let $e = c\Box < x_1, \ldots, x_n >$ and t a term, to

determine if e satisfies t ($t(e) = 1$) or not ($t(e) = 0$),we have to compute the inconsistency of a constraint:

- $t(e) = 0$ iff t, c is inconsistent,
- $t(e) = 1$ iff $\neg t, c$ is inconsistent,
- otherwise $t(e)$ is undetermined.

For example, let $t_1 = (temp \geq 15)$, $t_2 = (temp \leq 20)$ and $t_3 = (temp \geq 5)$: $t_1(e_6) = 0$, $t_2(e_6) = 1$ and $t_3(e_6)$ is undetermined,

Finally, as for missed values, $sim(e_i, e_j)$ can be defined by restricting the language to the set of terms t such that both $t(e_i)$ and $t(e_j)$ are defined. In example 2.2, object e_6 is highly similar to e_1 with respect to both $\mathcal{L}_1$ and $\mathcal{L}_2$.

3 Constraint Tree Grammars

We propose in this section a new formalism able to express a language specification: constraint tree grammars. Regular tree grammars are commonly used to specify language composed of structured terms. Such grammars [GS84] are defined by

- a set of constructors $\mathcal{F}$ (function symbols and their arity) such as $=$, $\geq$, or, red, $color$, ... We denote $T(\mathcal{F})$ the set of terms over $\mathcal{F}$ (for example $or(= (color, red), \geq (temp, 10))$),
- a set of non-terminal symbols containing an axiom noted A,
- and a set of production rules.

The set of production rules specify how terms can be built. In order to handle context dependent informations, we propose to extend this formalism by adding constraints to the production rules. In most constraint languages [MS98], constraints are defined over a structure $\mathcal{D}$ composed of a set of function symbols, an interpretation domain D, a set of constraint predicate symbols and a set of evaluation functions: these functions define the evaluation of terms on D and the boolean values of constrained atoms.

The idea is to use constraints to declaratively define biases on the language. This formalism has been deeply studied in [Moa00]. The main advantage here is the compositionality: from different languages (specified by their grammar), we can build a new grammar representing the union or intersection of initial languages.

Moreover production rules can be used to stochastically generate terms of the language for building a finite subset of the (possibly infinite) language specified by the grammar. Finally, weights can be associated with each rules in order to specify different probabilities.

3.1 Formal Definition

Formally, a constraint tree grammar is defined as follows:

Definition 2. *A* **Constraint Tree Grammar** *is a tuple* $G = (A, N, \mathcal{F}, \mathcal{D}, R)$ *where:*
- A *is the axiom,*
- N *is a set of non-terminal symbols* $X(x_1, x_2, \ldots, x_n)$*, where* $(x_1, x_2, \ldots, x_n)$ *represents the set of attributes linked to the symbol* X*.* $\tilde{N}$ *represents the set of symbols of* N*, without their attributes specifications* ($A \in \tilde{N}$)*.*
- $\mathcal{F}$ *is a set of terminal symbols, with their arities,*
- $\mathcal{D}$ *is constraint domain,*
- R *is a finite set of production rules* $X \to \alpha \Box c$*, where* $X \in \tilde{N}$*,* $\alpha \in T(\mathcal{F} \cup \tilde{N})$*, and* c *is a set of constraints on the attributes of the non-terminals occurring in the rule. This set is interpreted as a conjunction of constraints.*

For instance, the language composed of disjunctions of terms in $\mathcal{L}_2 \cup \mathcal{L}_3$ (§2.2) with at most two disjunctions is specified by the grammar in Table 1.

We use the classical notation of attribute grammars: the value of the attribute a in the non-terminal N is denoted by $N.a$, and if there are more than one occurrence of the same non-terminal in a rule, these occurrences are numbered from left to right, starting from 0.

Table 1. A Constraint Tree Grammar.

$$
\begin{array}{l}
G_c = \{\ A, \\
\quad \{A(d), B(d), T, Temp, Col\}, \\
\quad \{or/_2, =/_2, \leq/_2, temp, col, red, pink, orange, blue, 32, 212\}, \\
\quad (\ \mathbb{N}, \{+, 0, 1\}, \{=\}), \\
\quad \{ \\
\quad A \quad \to B\ \Box\{A.d \leq 2, A.d = B.d\} \\
\quad B_0 \quad \to or(B_1, B_2)\ \Box\{B_0.d = 1 + B_1.d + B_2.d\} \\
\quad B \quad \to T\ \Box\{B.d = 0\} \\
\quad T \quad \to \leq (temp, Temp) \\
\quad T \quad \to = (col, Col) \\
\quad Col \quad \to red \mid pink \mid blue \mid orange \\
\quad Temp \to 32 \mid 212 \quad \} \qquad\qquad \}
\end{array}
$$

In this grammar, $Temp \to 32 \mid 212$ is an abbreviation for the two rules $Temp \to 32$ and $Temp \to 212$ (we omit the constraint when it is $true$). The non-terminal symbol A has an attribute d indicating the number of disjunctions in a term. This number must be less or equal than 2 ($A.d \leq 2$).

3.2 Generated Language

We describe now the process of generating a term from a constraint tree grammar. The starting point for this process is made of the atom and an empty set of constraints $\emptyset$: $< A \Box \emptyset >$.

Then, a derivation step for a constrained term $< t \Box c >$ consists of choosing a non-terminal symbol X in t, choosing a derivation rule $X \to \alpha \Box c'$, substituting X with α in t to get t' and adding the constraint c' to c (the symbols A.n

occurring in the constraints are considered as variable in constraints; for this reason, these variables must be renamed in c').

In the previous example, the term $< A\Box\emptyset >$ can be derived in $< B\Box\{A.d < 2, A.d = B.d\} >$ using the first production rule. Then, this term can be derived either in $< or(B_1, B_2)\Box\{A.d \geq 2, A.d = B.d, B.d = B_1.d + B_2.d\} >$ using the second production rule, or in $< T\Box\{A.d \geq 2, A.d = B.d, B.d = 0\} >$ using the third one.

Definition 3. *Given a Constraint Tree Grammar $G = (A, N, \mathcal{F}, \mathcal{D}, R)$, the language generated from G (noted $L(G)$) is the set of ground terms (without non-terminal symbol) t such that there exists a constrained term $< t\Box c >$ obtained by derivations from $< A\Box\emptyset >$ and where the set of constraints c is satisfiable.*

Let us notice that when a term is generated, if the derivation leads to a terms $< t\Box c >$ such that c is not satisfiable, the derivation can be stopped (or backtrack) since no term of the language can be built by derivation from $< t\Box c >$.

For example, if we use the second rule from the term $< or(B_1, B_2)\Box\{A.d \geq 2, A.d = B.d, B.d = 1 + B_1.d + B_2.d\} >$, we can derive the term $< or(or(B_3, B_4), B_2)\ \Box\{A.d \geq 2, A.d = B.d, B.d = 1 + B_1.d + B_2.d, B_1.d = 1 + B_3.d + B_4.d\} >$. If the second rule is use again, we can obtain the term $< or(or(or(B_5, B_6), B_4), B_2)\ \Box\{A.d \geq 2, A.d = B.d, B.d = 1 + B_1.d + B_2.d, B_1.d = 1 + B_3.d + B_4.d, B_3.d = 1 + B_5.d + B_6.d\} >$ for which the set of constraints is unsatisfiable (the number of disjunctions was limited to 2 and this terms has 3 disjunctions).

3.3 Generating Random Values

The formalism of constraint tree grammars can only be used when the number of terminal symbol is finite. Most numeric domains are infinite and then the language $\mathcal{L}_{A \leq x} = \{Att_i \leq v_{i_j}, v_{i_j} \in [a_i, b_i\}$ can not be specified with such grammar. In this case, we propose to add one (or more) non-terminal symbol(s) *Rnd*: this non-terminal symbol will be derived in a randomly generated value. In our experimentations (in attribute-value case), we have considered mainly two ways for generating a values for an attribute Att_i, depending on two domains:

$D_{discrete}$**:** the values are selected from the set of possible values (appearing in the object descriptions)

$D_{continuous}$**:** the values are chosen in a range $[Min_i, Max_i]$ where Min_i (resp. Max_i) may be, for example, the minimum (resp. maximum) value for attribute Att_i in the set of objects.

4 Application to Classification

To test our approach, we apply our measure to classification: we compute the similarity between objects to classify and objects for which the class is known. Then the class of the most similar object is assigned (in case of equi-similar objects, the most frequent class is affected).

This evaluation has to answer some questions: do we need a great subset of the language, does this method is efficient? The Figure 2 shows the accuracy of the method on the mushrooms dataset with various languages: $\mathcal{L}_1$ is $\mathcal{L}_{A=v}$ with values generated by a random selection in the domain, $\mathcal{L}_2$ is $\mathcal{L}_{A=v}$ with values generated by randomly selecting an object and returning the corresponding attributes. $\mathcal{L}_2$, which gives a great weight to frequent values, has better accuracy. $\mathcal{L}_{AND}$ is $\mathcal{L}_2$ with conjunction and $\mathcal{L}_{OR}$ is $\mathcal{L}_2$ with disjunction. $\mathcal{L}_{AND}$ contains more specific terms than $\mathcal{L}_{OR}$ and its accuracy is better, but the best is $\mathcal{L}_2$.

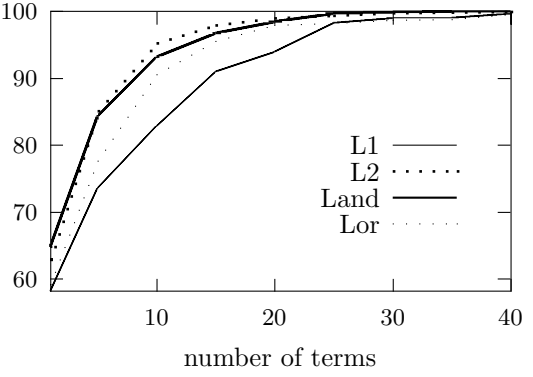

Fig. 2. Evolution of the accuracy with different languages

These tests consist of predicting the class of 5% (406 mushrooms) of the base using the 95% remaining (7719). Our classifier $\mathcal{L}^{class}$, written in C and running on an Ultra Sparc 5, takes 270 sec for this classification, generating 100 terms. Notice that the accuracy is 100% with 100 terms, even if $|\mathcal{L}_{A=v}| = 123$. Moreover, the 100% accuracy is obtained with an average of 32.5 terms using $\mathcal{L}_2$.

The following table shows the accuracy of $\mathcal{L}^{class}$ on the Glass problem with different languages: $\mathcal{L}^c_{A\leq x}$ (resp. $\mathcal{L}^d_{A\leq x}$) for continuous (resp. discrete) domains, in the same way $\mathcal{L}^c_{x\leq A\leq y}$ and $\mathcal{L}^d_{x\leq A\leq y}$ and we test $\mathcal{L}_{A=n}$ with discrete domains that can be considered because most values occur several times.

$\mathcal{L}^c_{A\leq x}$	$\mathcal{L}^d_{A\leq x}$	$\mathcal{L}^c_{x\leq A\leq y}$	$\mathcal{L}^d_{x\leq A\leq y}$	$\mathcal{L}_{A=n}$
75.04%	80.26%	75.5%	76.13%	49.39%

These results are obtained from 50 runs, with 10% of objects to classify, and with 1000 terms (one run = 1 second). Notice that 80% accuracy is one of the best results on the Glass problem. Moreover, we test with $\mathcal{L}^c_{A\leq x} \cup \mathcal{L}^c_{x\leq A\leq y}$ and get 77.39% accuracy. This example shows that results with an union of languages can be better than results with languages taken separately.

The Table 2 compares average accuracies on some domains of the UCI repository [BM98], for different well-known classifiers. These results are extract from [LM98].

The accuracy is measured by a 10-fold cross-validation. For $\mathcal{L}^{class}$, the accuracy is the average measure over 50 runs, with a random cross-validation (10%

Table 2. Comparison of accuracy on different domains.

Domains	$\mathcal{L}^{class}$	SCOPE	RISE	PEBLS	C4.5	CN2	SE-Learn
Echocardiogram	82.2	69.3	62.7	63.7	68.0	70.1	67.8
Glass	80.2	74.3	72.0	69.7	63.3	62.1	71.4
Hepatitis	78.0	78.7	76.9	82.4	81.6	81.2	-
Iris	94.6	94.7	95.3	95.5	95.8	94.0	96.0
Wine	96.1	95.5	98.9	97.7	94.9	92.7	98.8
Zoology	96.0	94.0	96.0	95.5	88.6	90.0	95.0

testing set), 1000 terms, and equivalent languages ($\mathcal{L}_{A\leq x}$ for numeric attributes, and $\mathcal{L}_{A=v}$ for symbolic ones).

Our system has good results over several domains, like Echocardiogram, even with a basic language and a "naive" nearest-neighbor algorithm. We are currently studying some improvements: using a K-NN algorithm, weighting terms in the language, ...

Complexity: The algorithm requires a generation function and an evaluation function associated with the language. We have chosen to represent the language with a grammar where each rule is associated with a probability to be selected: the time needed for the generation of terms is insignificant. Let N be the number of known objects and let M be the number of objects to classify.

We propose to compute a matrix storing the distance between each object to classify and each known object: the size of this matrix is $M \times N$. Let T be the number of terms generated. For each term, we have to compute $M + N$ evaluations and $N.M$ comparisons to compute the similarity. Then the time complexity depends on $T.(a(M+N)+b(N.M))$ which is polynomial (a represents time for an evaluation, b represents time for a comparison). However, when $M.N$ is high this method requires a large amount of memory to store the distance matrix.

5 Conclusion and Further Works

We proposed in this paper a generic similarity measure based on a language specification, that allows to cope with various formalisms in a uniform way: attribute-values, first order logic, constraint FOL. The language allows to express relevant (or supposed relevant) properties of objects, and to easily introduce biases. The language is specified by a grammar that allows to generate terms.

For the moment, it has been implemented in the system $\mathcal{L}^{class}$ for attribute-value objects and for classification tasks.

The experiments show that the quality of classification task depend on the language used. For many practical domains, $\mathcal{L}^{class}$ gets comparable or better results than usual approaches with simple languages.

We now plan to test this similarity measure in a constraint first order logic framework and with unstrutured datas like texts.

References

[Aha89] D. Aha. Incremental, instance based-learning of independent and graded concept descriptions. In *Sixth International Machine Learning Workshop (ML89)*, pages 387–391, 1989.

[Bis92] G. Bisson. Learning in FOL with a similarity measure. In *11th National Conf. on Artificial Intelligence (AAAI), San Jose, CA.*, pages 82–87. AAAI Press, 1992.

[BM98] C.L. Blake and C.J. Merz. UCI repository of machine learning databases, 1998.

[CS93] S. Cost and S. Salzberg. A weighted nearest neighbor algorithm for learning with symbolic features. *Machine Learning ,10(1), 57–78*, 1993.

[DeG86] Morris H. DeGroot. *Probability and Statistics*. Addison-Wesley Series in Statistics. Addison-Wesley, Reading, MA, USA, 2nd edition, 1986.

[Dom95] P. Domingos. Rule induction and instance-based learning: A unified approach. In *Fourteenth International Joint Conference on Artificial Intelligence (IJCAI'95), Montreal, Canada*, pages 1226–1232. Morgan & Kaufmann, 1995.

[EW96] W. Emde and D. Wettschereck. Relational instance-based learning. In Saitta L., editor, *13th Int. Conf. on Machine Learning (ICML'96), Bari, Italy*, pages 122–130. Morgan & Kaufmann, 1996.

[GCM95] C. Giraud-Carrier and T. Martinez. An efficient metric for heterogeneous inductive learning applications in the attribute-value language. In *Proceedings of GWIC'94*, pages Vol. 1, 341–350. Kluwer Academic Publishers, 1995.

[GS84] F. Gécseg and M. Steinby. *Tree Automata*. Akadémiai Kidoó, Budapest, 1984.

[HWB01] Tamás Horváth, Stefan Wrobel, and Uta Bohnebeck. Relational instance-based learning with lists and terms. *Machine Learning*, 43(1/2):53–80, 2001.

[LM98] N. Lachiche and P. Marquis. Scope classification: An instance-based learning algorithm with a rule-based characterization. *Lecture Notes in Computer Science*, 1398:268–??, 1998.

[Moa00] F. Moal. *Langages de biais en Apprentissage Symbolique*. PhD thesis, LIFO, Université d'Orléans, France, December 2000.

[MS98] Kim Marriott and Peter J. Stuckey. *Programming with Constraints: An Introduction*. The MIT Press, 1998.

[MT94] T. Mohri and H. Tanaka. An optimal weighting criterion of case indexing for both numeric and symbolic attributes. In *Case-Based Reasoning Workshop*, pages 123–127. AAAI Press, 1994.

[Pla95] Enric Plaza. Cases as terms: A feature term approach to the structured representation of cases. In *ICCBR*, pages 265–276, 1995.

[Seb97] M. Sebag. Distance induction in first order logic. In *Proceedings of ILP'97*, pages 264–272. Springer-Verlag, 1997.

[SS94] M. Sebag and M. Schoenauer. *Topics in Case-Based Reasonning*, volume 837 of *LNAI*, chapter A Rule-based Similarity Measure, pages 119–130. Springer-Verlag, 1994.

[SW86] C. Stanfill and D. Waltz. Toward memory-based reasoning. *Communication of the ACM, 29(12):1213-1228*, 1986.