Découverte de mappings probabilistes entre taxonomies

Rémi Tournaire
LIG
University of Grenoble
Remi.Tournaire@imag.fr

Jean-Marc Petit
LIRIS
University of Lyon
jmpetit@liris.cnrs.fr

Marie-Christine Rousset
LIG
University of Grenoble
Marie-Christine.Rousset@imag.fr

Alexandre Termier
LIG
University of Grenoble
Alexandre.Termier@imag.fr

Résumé

Notre but est d'analyser une approche pour définir et découvrir des mappings probabilistes entre deux taxonomies. D'abord, nous comparons 2 façons de modéliser les mappings probabilistes qui sont compatibles avec les contraintes logiques déclarées dans les taxonomies. Nous suivons une démarche bayésienne pour estimer les probabilités des mappings, en exploitant les descriptions des instances associées aux classes des taxonomies. Ensuite, nous présentons un algorithme de type generate and test qui détermine les mappings dont la probabilité dépasse un seuil donné, en minimisant le nombre d'estimations de probabilités. Enfin, nous montrons les résultats d'une expérimentation poussée de notre approche sur des données synthétisées et contrôlées avec lesquelles nous avons effectué des mesures à la fois qualitatives et quantitatives.

Mots clefs : taxonomie, ontologie, alignement, mapping, logique, probabilités

1 Introduction

The decentralized nature of the development of Web data management systems makes inevitable the independent construction of a large amount of personalized taxonomies used for annotating data and resources at Web scale. Taxonomies are hierarchical structures appropriate to data categorization and semantic annotation of resources. They play a prominent role in the Semantic Web since they are central components of OWL [8] or RDF(S) [23] ontologies. A taxonomy constrains the vocabulary used to express metadata or semantic annotations to be classes that are related by structural relationships. Taxonomies are easy to create and understand by humans while being machine interpretable and processable thanks to a formal logical semantics supporting reasoning capabilities.

In this setting, establishing *semantic mappings* between taxonomies is the key to enable collaborative exchange of semantic data. Manually finding such mappings is clearly not possible at the Web scale. Therefore, the automatic discovery of semantic mappings is the bottleneck for scalability purposes.

Many techniques and prototypes have been developed to suggest candidate mappings between several knowledge representations including taxonomies, ontologies or schemas (see [31, 32]

for surveys). Most of the proposed approaches rely on evaluating the degree of similarity between the elements (e.g., classes, properties, instances) of one ontology and the elements of another ontology. Many different similarity measures are proposed and often combined. Most of them are based on several syntactic, linguistic or structural criteria to measure the proximity of the terms used to denote the classes and/or their properties within the ontology. Some of them exploit characteristics of the data declared as instances of the classes (e.g. [12]).

Almost all the existing matching systems return for every candidate pair of elements a coefficient in the range [0,1] which denotes the strength of the semantic correspondence between those two elements ([18, 27, 5]). Those coefficients are the basis for yearly international comparative evaluation campaigns [16]. Those approaches usually consider each candidate mapping in isolation. In particular, they do not take into account possible logical implications between mappings, which can be inferred from the logical inclusion axioms declared between classes within each ontology. This raises a crucial issue : the similarity coefficients returned by the existing ontology or schema matching systems cannot be interpreted as *probabilities* of the associated mappings. On the other hand, some approaches for detecting semantic mappings by logical reasoning have been proposed (e.g., [21]). By construction, such logical methods are limited to discover mappings that are *certain*.

We claim that *uncertainty* is intrinsic to mapping discovery. It is first due to the methods employed for detecting them. Another important reason is that the mappings are usually interpreted as simple semantic relations such as subsomption or equivalence relations between classes, which is often an oversimplification of the complex overlapping relation existing in the reality between two classes of different and independenty developed ontologies.

In this paper, we propose an approach to discover automatically *probabilistic mappings* between taxonomies of classes.

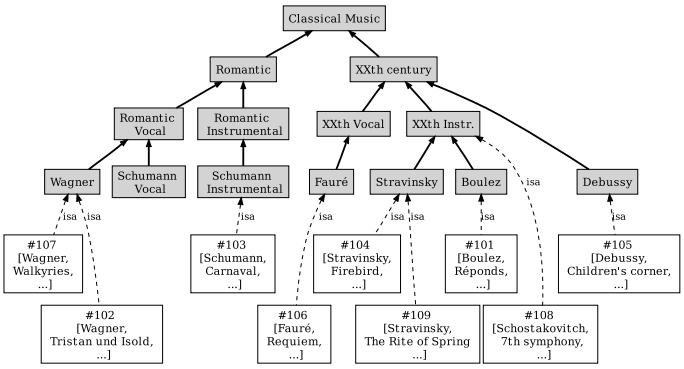
First, we investigate and compare two ways of modeling probabilistic mappings which are compatible with the logical constraints declared in each taxonomy. In those two probabilistic models, the probability of a mapping relies on the joint probability distribution of the involved classes. They differ on the property of *monotonicity* of the corresponding probability function with respect to the logical implication.

For estimating the mappings probabilities, we follow a Bayesian approach to statistics by exploiting the description of the *instances* categorized in each taxonomy as observations for the involved classes. The point is that to estimate the joint probability distribution of two classes C_1 and C_2 of different taxonomies, we have to determine among the instances that are declared in C_1 the ones that can be classified in C_2 (based on their description), and similarly for the classification in C_2 of instances that are declared in C_1. Different classifiers can be used for that purpose.

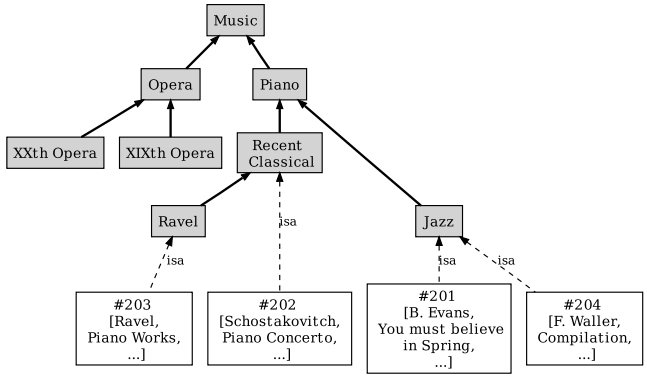
Based on the above probabilistic setting, we have designed, implemented and experimented a *generate and test* algorithm for discovering the mappings whose probability is greater than a given threshold. In this algorithm, the monotonicity of the probability function is exploited for avoiding the probability estimation of as many mappings as possible.

We have performed thorough experiments on controlled synthetic data to measure the performances of such a systematic approach in fonction of the number of possible mappings and the number of instances per classes. We have also performed qualitative experiments to measure the impact of the classifiers used to estimate the probabilities on the precision and recall of the mappings returned by our algorithm.

The paper is organized as follows. Section 2 presents the formal background and states the problem considered in this paper. Section 3 is dedicated to the definition and computation of mapping probabilities. In Section 4, we present the algorithm that we propose for discovering mappings with high probabilities (i.e., greater than a threshold). Section 5 surveys the quantitative and qualitative experiments that we have done on synthetic controlled data. Finally, in



(a) Taxonomy $\mathcal{T}_1$



(b) Taxonomy $\mathcal{T}_2$

FIG. 1 – 2 Taxonomies and associated instances

Section 6, we compare our approach to existing works and we conclude.

2 Formal background

We first define taxonomies as a graphical notation and its interpretation in the standard first-order logical semantics, on which the inheritance of instances is grounded. Then, we define *mappings* between taxonomies as inclusion statements between classes of two different taxonomies. Finally, we set the problem statement of matching taxonomies that we consider in this paper.

2.1 Taxonomies : classes and instances

Given a vocabulary $\mathcal{V}$ denoting a set of classes, a *taxonomy* $\mathcal{T}_\mathcal{V}$ is a Directed Acyclic Graph (DAG) where each node is labelled with a distinct *class* name of $\mathcal{V}$, and each arc between a node labelled with C and a node labelled by D represents a *specialization relation* between the classes C and D.

Each class in a taxonomy can be associated with a set of *instances* which have an *identifier* and a content *description* modeled with an attribute-value language.

By a slight abuse of notation, we will speak of the instance i to refer to the instance identified by i.

Figure 1 shows two samples of taxonomies related to the Music domain. Bold arrows are used for representing specialization relations between classes, and dashed arrows for membership relation between instances and classes. In both taxonomies, some instances, with attribute-value description denoted between brackets, are associated to classes. For example, #102 is an instance identifier and [Wagner, Tristan und Isold, ...] its associated description.

The instances that are in the scope of our data model can be varied : they can be web pages (whose content description is a set of words) identified by their URL, RDF resources (whose content description is a set of RDF triples) identified by a URI, or audio or video files identified by a signature and whose content description may be attribute-value metadata that can be extracted from those files.

We consider only boolean attribute-value description. Such a description could be obtained by discretization of attribute-value pairs given in a more complex language, like in Figure 1 where textual values are used. We consider that, possibly after a preprocessing which is out of the scope of this paper, the instances are described in function of a fixed set of boolean attributes $\{At_1, \ldots, At_m\}$. Then, for an instance i, its description, denoted *descr(i)*, is a vector $[a_1, \ldots, a_m]$ of size m such that for every $j \in [1..m]$, $a_j = 1$ if the attribute At_j belongs to the content description of i, and $a_j = 0$ otherwise.

Taxonomies have a logical semantics which provides the basis to define formally the extension of a class as the set of instances that are declared or can be *inferred* for that class.

2.2 Logical semantics

There are several graphical or textual notations for expressing the specialization relation between a class C and a class D in a taxonomy. For example, in RDF(S) [23] which is the first standard of the W3C concerning the Semantic Web, it is denoted by $(C\ rdfs:subclassOf\ D)$. It corresponds to the inclusion statement $C \sqsubseteq D$ in the description logics notation.

Similarly, a membership statement denoted by an *isa* arc from an instance i to a class C corresponds in the RDF(S) notation to $(i\ rdf:type\ C)$, and to $C(i)$ in the usual notation of description logics.

All those notations have a standard model-theoretic logical semantics based on interpreting classes as sets : an *interpretation* $\mathcal{I}$ consists of a non empty domain of interpretation $\Delta^{\mathcal{I}}$ and a function $.^{\mathcal{I}}$ that interprets each class as a non empty subset of $\Delta^{\mathcal{I}}$, and each instance identifier as an element of $\Delta^{\mathcal{I}}$. The classes declared in a taxonomy are interpreted as non empty subsets because they are object containers. According to the *unique name assumption*, two distinct identifiers a and b have a distinct interpretation ($a^{\mathcal{I}} \neq b^{\mathcal{I}}$) in any interpretation $\mathcal{I}$.

$\mathcal{I}$ is a model of a taxonomy $\mathcal{T}$ if :

- for every inclusion statement $E \sqsubseteq F$ of $\mathcal{T}$: $E^{\mathcal{I}} \subseteq F^{\mathcal{I}}$,
- for every membership statement $C(a)$ of $\mathcal{T}$: $a^{\mathcal{I}} \in C^{\mathcal{I}}$.

An inclusion $G \sqsubseteq H$ is *inferred* by a taxonomy $\mathcal{T}$ (denoted by $\mathcal{T} \models G \sqsubseteq H$) iff in every model $\mathcal{I}$ of $\mathcal{T}$, $G^{\mathcal{I}} \subseteq H^{\mathcal{I}}$.

A membership $C(e)$ is *inferred* by $\mathcal{T}$ (denoted by $\mathcal{T} \models C(e)$) iff in every model $\mathcal{I}$ of $\mathcal{T}$, $e^{\mathcal{I}} \in C^{\mathcal{I}}$.

Let $\mathcal{D}$ be the set of the instances associated to a taxonomy $\mathcal{T}$. The *extension* of a class C in $\mathcal{T}$, denoted by $Ext(C, \mathcal{T})$, is the set of instances for which it can be inferred from the membership and inclusion statements declared in the taxonomy that they are instances of C :

$Ext(C, \mathcal{T}) = \{d \in \mathcal{D} /\ \mathcal{T} \models C(d)\}$

2.3 Mappings

The mappings that we consider are inclusion statements involving classes of two different taxonomies $\mathcal{T}_1$ and $\mathcal{T}_2$. To avoid ambiguity and without loss of generality, we consider that each taxonomy has its own vocabulary : by convention we index the names of the classes by the index of the ontology to which they belong. For instance, when involved in a mapping, the class $XXth\ Opera$ of the taxonomy $\mathcal{T}_2$ of Figure 1 will be denoted by $XXth\ Opera_2$ while the class $XXth\ Vocal$ of the taxonomy $\mathcal{T}_1$ will be denoted by $XXth\ Vocal_1$.

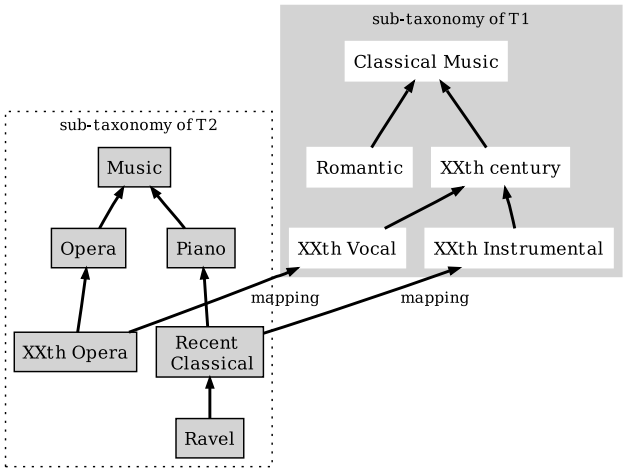


FIG. 2 – 2 mappings between $\mathcal{T}_1$ and $\mathcal{T}_2$

Mappings between $\mathcal{T}_1$ and $\mathcal{T}_2$ are of the form $A_1 \sqsubseteq B_2$ or $A_2 \sqsubseteq B_1$ where A_1 and B_1 denote classes of $\mathcal{T}_1$ and A_2 and B_2 denote classes of $\mathcal{T}_2$.

For a mapping m of the form $A_i \sqsubseteq B_j$, its left-hand side A_i will be denoted $lhs(m)$ and its right-hand side will be denoted $rhs(m)$.

A mapping $A_i \sqsubseteq B_j$ has the same meaning as a specialization relation between the classes A_i and B_j, and thus is interpreted in logic in the same way, as a set inclusion. The logical entailment between classes extends to logical entailment between mappings as follows.

Definition 1 (Entailment between mappings) *Let $\mathcal{T}_i$ and $\mathcal{T}_j$ be two taxonomies. Let m and m' be two mappings between $\mathcal{T}_i$ and $\mathcal{T}_j$: m entails m' (denoted $m \preceq m'$) iff every model of $\mathcal{T}_i$, $\mathcal{T}_j$ and m is also a model of m'.*

It is straightforward to show that $\preceq$ is a (partial) order relation on the set $\mathcal{M}(\mathcal{T}_i, \mathcal{T}_j)$ of mappings between the two taxonomies $\mathcal{T}_i$ and $\mathcal{T}_j$. If $m \preceq m'$, we will say that m is more specific than m' (also that m entails m') and that m' is more general than m (also that m' is an implicate of m).

The following proposition characterizes the logical entailment between mappings in function of the logical entailment between the classes of their left hand sides and right hand sides.

Proposition 1 *Let m and m' be two mappings between two taxonomies. Let $\mathcal{T}_i$ be the taxonomy of $lhs(m)$, and $\mathcal{T}_j$ the taxonomy of $rhs(m)$.*

$m \preceq m'$ iff

- $lhs(m)$ and $lhs(m')$ are classes of the same taxonomy $\mathcal{T}_i$

and

- $\mathcal{T}_i \models lhs(m') \sqsubseteq lhs(m)$ and $\mathcal{T}_j \models rhs(m) \sqsubseteq rhs(m')$

For example, two mappings between taxonomies $\mathcal{T}_1$ and $\mathcal{T}_2$ of Figure 1 are illustrated in Figure 2 :

- the mapping $XXth\ Opera_2 \sqsubseteq XXth\ Vocal_1$ is more specific than the mapping $XXth\ Opera_2 \sqsubseteq XXth\ Century_1$,
- and the mapping $RecentClassical_2 \sqsubseteq XXth\ Instrumental_1$ is more specific than the mapping $Ravel_2 \sqsubseteq Classical\ Music_1$.

2.4 Problem statement

Among all possible mappings between two taxonomies, we want to determine those that are the most *probable* given the descriptions of the instances associated to each class of the taxonomies. More precisely, the *main problem* addressed in this paper is the design of an efficient *generate and test* algorithm which minimizes the number of calls to the probability estimator for determining those mappings whose probability exceeds a certain threshold. The mappings returned by this algorithm will be said *probabilistically valid* (*valid* for short).

Two subproblems are emphasized. The *first subproblem* to handle is the choice of an appropriate *probabilistic model* for defining the probability of a mapping. As mentioned in the introduction, a probabilistic semantics of mappings cannot be independent of the logical semantics. In particular, it is expected that a mapping logically entailed by a mapping with a high probability (i.e., whose probability exceed a threshold) will also get a high probability. The *second subproblem* is then to find a good *probability estimator* to compute mapping probabilities, given two taxonomies and the description of their instances.

3 Mapping probabilities : models and estimation

3.1 Modeling probabilistic mappings

We have considered two relevant probabilistic models for modeling uncertain mappings. They are both based on the discrete probability measure defined on subsets of the sample set representing the set of all possible instances of the two taxonomies. From now on, we will denote $Pr(E)$ the probability for an instance to be an element of the subset E.

The first model defines the probability of a mapping $A_i \sqsubseteq B_j$ as the conditional probability for an instance to be an instance of B_j knowing that it is an instance of A_i. It is the natural way to extend the logical semantics of entailment to probabilities.

The second model comes directly from viewing classes as subsets of the sample space : the probability of $A_i \sqsubseteq B_j$ is the probability for an element to belong to the set $\overline{A_i} \cup B_j$, where $\overline{A_i}$ denotes the complement set of A_i in the sample set.

These two models are described in the following definition.

Definition 2 (Two probabilities for a mapping) *Let m be a mapping of the form $A_i \sqsubseteq B_j$.*

- *Its* conditional probability*, denoted $P_c(m)$, is defined as : $P_c(m) = Pr(B_j|A_i)$.*
- *Its* union_of_set probability*, denoted $P_u(m)$, is defined as : $P_u(m) = Pr(\overline{A_i} \cup B_j)$.*

The following proposition states the main (comparative) properties of those two probabilistic models. In particular, they both meet the logical semantics for mappings that are certain, and they can both be equivalently expressed using joint probabilities.

Proposition 2 *Let m be a mapping between two taxonomies $\mathcal{T}_i$ and $\mathcal{T}_j$. The following properties hold :*

1. $P_u(m) \geq P_c(m)$.
2. *If m is a* certain mapping *(i.e., $\mathcal{T}_i \mathcal{T}_j \models m$) :*
   $P_c(m) = P_u(m) = 1$.
3. $P_u(m) = 1 + Pr(lhs(m) \cap rhs(m)) - Pr(lhs(m))$
4. $P_c(m) = \frac{Pr(lhs(m) \cap rhs(m))}{Pr(lhs(m))}$

They differ on the monotonicity property w.r.t the (partial) order $\preceq$ corresponding to logical implication (cf. Definition 1) : P_u verifies a property of monotonicity whereas P_c verifies a property of *weak* monotonicity as stated in the following theorem.

Theorem 1 (Property of monotonicity) *Let m and m' two mappings.*

1. *If $m \preceq m'$ then $P_u(m) \leq P_u(m')$*
2. *If $m \preceq m'$ and $lhs(m) = lhs(m')$ then $P_c(m) \leq P_c(m')$*

The proof [34] results from Proposition 1 and Proposition 2 which relate mappings with the classes of their left hand sides and right hand sides for logical entailment and probabilities respectively, and from considering (declared or inherited) class inclusions within each taxonomy as statements whose probability is equal to 1.

3.2 Bayesian estimation of mappings probabilities

As shown in Proposition 2, the computation of $P_u(m)$ and $P_c(m)$ relies on computing the set probability $Pr(lhs(m))$ and the joint set probability $Pr(lhs(m) \cap rhs(m))$. Those values are unknown and must be estimated. They are the (unknown) parameters of the underlying Bernoulli distributions modeling the membership function to a set as a random variable taking only two possible values 0 or 1. Following the Bayesian approach to statistics [9], we model those (unknown) parameters as continuous random variables, and we use *observations* to infer their *posterior* distribution from their *prior* distribution. In the absence of any particular knowledge, the prior distribution is usually set to the uniform distribution. In probability theory, a natural way of estimating the value of a parameter modeled by a random variable is to take its *expected value*. All this is summarized in the following definition.

Definition 3 (Bayesian estimator of $Pr(E)$)
Let E be a subset of the sample set Ω. Let $\mathcal{O}$ be a sample of observed elements for which it is known whether they belong or not to E. The Bayesian estimator of $Pr(E)$, denoted $\widehat{Pr}(E)$, is the expected value *of the posterior distribution of $Pr(E)$ knowing the observations on the membership to E of each element in $\mathcal{O}$, and setting the prior probability of a random set to $\frac{1}{2}$, and of the intersection of two random sets to $\frac{1}{4}$.*

Setting the prior probabilities to $\frac{1}{2}$ and $\frac{1}{4}$ depending on whether E is a class or a conjunction of classes corresponds to the uniform distribution of instances among the classes.

Let $\widehat{Ext}(E, \mathcal{O})$ be the set of observed instances of $\mathcal{O}$ that are recognized to be instances of E. According to a basic theorem in probability theory (Theorem 1, page 160, [9]), if the prior distribution of the random variable modeling $Pr(E)$ is a *Beta distribution* of parameters α and β, then its posterior distribution is also a Beta distribution the parameters of which are : $\alpha + |\widehat{Ext}(E, \mathcal{O})|$ and $\beta + |\mathcal{O}|$.

The Beta distribution is a family of continuous probability distributions parameterized by two parameters α and β which play an important role in Bayesian statistics. If its two parameters are equal to 1, it corresponds to the uniform distribution for the associated random variable. Its *expected value* is : $\frac{\alpha}{\alpha+\beta}$.

In our setting, the set $\mathcal{O}$ is the union of the two (possibly disjoint) sets $\mathcal{O}_i$ and $\mathcal{O}_j$ of instances observed in two distinct taxonomies $\mathcal{T}_i$ and $\mathcal{T}_j$. This raises the issue of computing the set $\widehat{Ext}(E, \mathcal{O}_i \cup \mathcal{O}_j)$, specially when E is the conjonction of a class C_i of the taxonomy $\mathcal{T}_i$ and a class D_j of the other taxonomy $\mathcal{T}_j$. In this case :
$\widehat{Ext}(C_i \cap D_j, \mathcal{O}_i \cup \mathcal{O}_j) = \widehat{Ext}(C_i, \mathcal{O}_i \cup \mathcal{O}_j) \cap \widehat{Ext}(D_j, \mathcal{O}_i \cup \mathcal{O}_j)$

Since the two taxomomies have been created and populated independently by different users, the only information that can be extracted from those two taxonomies are the extensions of each class *within* each taxonomy : $Ext(C_i, \mathcal{T}_i)$ and $Ext(D_j, \mathcal{T}_j)$.

By construction, it is likely that their intersection contains very few instances or even no instance at all. Therefore, we use *automatic classifiers* to compute $\widehat{Ext}(E, \mathcal{O})$. The machine learning community has produced several types of classifiers that have been extensively (theoretically and experimentally) studied (see [29] for a survey) and have been made available through open source platforms like Weka [36]. They are based on different approaches (e.g., Naive Bayes learning, decision trees, SVM) but they all need a training phase on two sets of positive and negative examples. Let $\mathcal{C}$ be a classifier. Let E be a class of one of the two taxonomies that we denote by $\mathcal{T}_i$, the other one being denoted $\mathcal{T}_j$. For computing $\widehat{Ext}(E, \mathcal{O})$ we follow the same approach as [12] :

- $\mathcal{C}$ is trained on the descriptions of the elements of the two sets $Ext(E, \mathcal{T}_i)$ and $\mathcal{O}_i \backslash Ext(E, \mathcal{T}_i)$ taken as the sets of positive and negative examples respectively,
- $\mathcal{C}$ is then applied to each instance of $\mathcal{O}_j$ to recognize whether it belongs to E or not.

As a result, the following theorem provides a simple way to compute the Bayesian estimations $\widehat{P_u}(m)$ and $\widehat{P_c}(m)$ of the two probabilities $P_u(m)$ and $P_c(m)$ defined in Definition 2.

Theorem 2 (Estimation of mapping probabilities) *Let $m : C_i \sqsubseteq D_j$ be a mapping between two taxonomies $\mathcal{T}_i$ and $\mathcal{T}_j$. Let $\mathcal{O}$ be the union of instances observed in $\mathcal{T}_i$ and $\mathcal{T}_j$. Let $N = |\mathcal{O}|$, $N_i = |\widehat{Ext}(C_i, \mathcal{O})|$, $N_j = |\widehat{Ext}(D_j, \mathcal{O})|$ and $N_{ij} = |\widehat{Ext}(C_i \cap D_j, \mathcal{O})|$.*

- $\widehat{P_u}(m) = 1 + \frac{1+N_{ij}}{4+N} - \frac{1+N_i}{2+N}$
- $\widehat{P_c}(m) = \frac{1+N_{ij}}{4+N} \times \frac{2+N}{1+N_i}$

It is worth comparing the (Bayesian) ratios $\frac{1+N_i}{2+N}$ and $\frac{1+N_{ij}}{4+N}$ appearing in the formulas for computing $\widehat{P_u}(m)$ and $\widehat{P_c}(m)$ in Theorem 2 with the corresponding ratios $\frac{N_i}{N}$ and $\frac{N_{ij}}{N}$ that would have been obtained by following the standard (frequency-based) approach of statistics (as it is the case for instance in [12]). The corresponding ratios converge to the same expected value when there are many instances, but the Bayesian ratios are more robust to a small number of instances. In contrast with the frequency-based approach, they are defined even in the case where no instance is observed : their respective values (i.e.,$\frac{1}{2}$ and $\frac{1}{4}$) in this particular case correspond to the probability of random sets and the joint probability of two random sets respectively for a uniform distribution of instances in the sample set.

4 Generate and test of candidate probabilistic mappings

Given two taxonomies $\mathcal{T}_i$ and $\mathcal{T}_j$ and their associated instances, the goal is to determine all mappings m of $\mathcal{M}(\mathcal{T}_i, \mathcal{T}_j)$ verifying a probabilistic-based criterion of validity that will be denoted by $\widehat{P}(m) \geq S$.

$\widehat{P}(m) \geq S$ is a parameter in the algorithm, which can be one of the three following validity criteria :

- *Validity criterion 1* : $\widehat{P_u}(m) \geq S_u$
- *Validity criterion 2* : $\widehat{P_c}(m) \geq S_c$
- *Validity criterion 3* : $\widehat{P_c}(m) \geq S_c$ and $\widehat{P_u}(m) \geq S_u$

where S_u and S_c are two thresholds in $[0; 1]$.

Candidate mapping generation

The principle of the algorithm is to generate candidate mappings from the classes in the two taxonomies partitioned into levels obtained by *a reverse topological ordering* [6].

Topolological levels

For a DAG $\mathcal{T}$, the topological levels obtained by reverse topological ordering are inductively defined as follows :

- $Level(0, \mathcal{T})$ is the set of nodes with no outgoing arc.

- $Level(i, \mathcal{T})$ is the set of nodes which are connected by outgoing arcs only to nodes belonging to levels strictly lower than i.

For example, the classes in the taxonomy $\mathcal{T}_2$ of Figure 1 are partitioned into 4 levels :

- $Level(0, \mathcal{T}_2) = \{Music\}$
- $Level(1, \mathcal{T}_2) = \{Opera, Piano\}$
- $Level(2, \mathcal{T}_2) = \{XXth\ Opera, XIXth\ Opera,$
$Recent\ Classical, Jazz\}$
- $Level(3, \mathcal{T}_2) = \{Ravel\}$

Generating candidate mappings by levels

Let $\mathcal{T}_i$ and $\mathcal{T}_j$ two taxonomies. Let nb_i and nb_j be the number of levels obtained by their respective reverse topological ordering. Let k such that $0 \leq k \leq nb_i + nb_j$. The set of *generated candidate mappings of level* k, denoted by $GenMapLevel(k, \mathcal{T}_i, \mathcal{T}_j)$, is obtained as follows :

$$GenMapLevel(k, \mathcal{T}_i, \mathcal{T}_j) =$$

$$\bigcup_{n+m=k,\ 0\leq n\leq nb_i,\ 0\leq m\leq nb_j} \{A_i \sqsubseteq A_j | A_i \in Level(nb_i - n, \mathcal{T}_i)\ ,\ A_j \in Level(m, \mathcal{T}_j)\}$$

$$\cup \bigcup_{n+m=k,\ 0\leq n\leq nb_j,\ 0\leq m\leq nb_i} \{B_j \sqsubseteq B_i | B_j \in Level(nb_j - n, \mathcal{T}_j)\ ,\ B_i \in Level(m, \mathcal{T}_i)\}$$

For the taxonomies of Figure 1, $GenMapLevel(0, \mathcal{T}_1, \mathcal{T}_2)$ is constituted of the following mappings :

-$Wagner_1 \sqsubseteq Music_2$
-$Schumann\ Vocal_1 \sqsubseteq Music_2$
-$Schumann\ Instrumental_1 \sqsubseteq Music_2$
-$Faure_1 \sqsubseteq Music_2$
-$Stravinsky_1 \sqsubseteq Music_2$
-$Boulez_1 \sqsubseteq Music_2$
-$Ravel_2 \sqsubseteq Classical\ Music_1$

We only give two of the nine mappings of $GenMapLevel(1, \mathcal{T}_1, \mathcal{T}_2))$:

- $Recent\ Classical_2 \sqsubseteq Classical\ Music_1$
- $XXth\ Instrumenta_2 \sqsubseteq Music_2$

The following proposition is a corollary of Proposition 1.

Proposition 3 *Let $\mathcal{T}_i$ and $\mathcal{T}_j$ two taxonomies.*
Let $\mathcal{M}(\mathcal{T}_i, \mathcal{T}_j)$ be the set of all possible mappings.

1. *The mappings in $GenMapLevel(0, \mathcal{T}_i, \mathcal{T}_j)$ are the most general mappings of $\mathcal{M}(\mathcal{T}_i, \mathcal{T}_j)$ for the entailment relation $\preceq$.*
2. *Let $m \in GenMapLevel(k, \mathcal{T}_i, \mathcal{T}_j)$, and let m' be a mapping in $\mathcal{M}(\mathcal{T}_i, \mathcal{T}_j)$ which entails m, then there exists $k' > k$ such that $m' \in GenMapLevel(k', \mathcal{T}_i, \mathcal{T}_j)$.*
3. *The set of subsets $GenMapLevel(k, \mathcal{T}_i, \mathcal{T}_j)$ ($0 \leq k \leq nb_i + nb_j$) is a partition of $\mathcal{M}(\mathcal{T}_i, \mathcal{T}_j)$.*

Pruning the candidate mappings to test

Based on the monotonicity property of the probability function P_u (Theorem 1), every mapping m' that entails a mapping m such that $P_u(m) < S_u$ verifies $P_u(m') < S_u$. Therefore, in the algorithm, if the validity criterion involves $\widehat{P_u}$, we prune the probability estimation of all mappings that entail any m such that $\widehat{P_u}(m) < S_u$. We shall use the notation $Entailing(m)$ to denote the set of all mappings that entails m. Similarly, based on the property of weak monotonicity of the probability function P_c (Theorem 1), if the validity criterion involves $\widehat{P_c}$, when a tested candidate mapping m is such that $\widehat{P_c}(m) < S_c$ we prune the probability estimation of all mappings that entail m having the same left-hand side as m. We shall denote this set : $Implicants_c(m)$.

Based on Proposition 1, $Implicants(m)$ and $Implicants_c(m)$ can be generated from $\mathcal{T}_i$ and $\mathcal{T}_j$.

Proposition 3 shows that testing the validity of mappings following the levels induced by reverse topological ordering of the two input taxonomies maximizes the number of pruning.

The resulting generate and test algorithm is described in Algorithm 1, in which :

- $\widehat{P}(m) \geq S$ in line 5 denotes a generic validity criterion that can be instantiated either by $\widehat{P_u} \geq S_u$, or by $\widehat{P_c} \geq S_c$, or by $\widehat{P_c} \geq S_c$ and $\widehat{P_u} \geq S_u$.
- In the case where the validity criteria involves $\widehat{P_c}$, $Implicants(m)$ in Line 8 must be replaced by $Implicants_c(m)$.
- In Line 2, nb_{levels} is the sum $nb_i + nb_j$ of the levels produced by a reverse topological order applied to the two taxonomies $\mathcal{T}_i$ and $\mathcal{T}_j$.

Algorithm 1 Generate and test all candidate mappings

Require: $\mathcal{T}_i, \mathcal{T}_j, nb_{levels}$, threshold S
Ensure: return $\{m \in \mathcal{M}(\mathcal{T}_i, \mathcal{T}_j) / \widehat{P}(m) \geq S\}$

```
1:  M_Val ← ∅, M_NVal ← ∅
2:  for k = 0 to nb_levels do
3:     for each m ∈ GenMapLevel(k, T_i, T_j) do
4:        if m ∉ M_NVal then
5:           if P̂(m) ≥ S then
6:              M_Val ← M_Val ∪ {m}
7:           else
8:              M_NVal ← M_NVal ∪ Implicants(m)
9:           end if
10:       end if
11:    end for
12: end for
13: return M_Val
```

5 Experiments

For the purpose of systematic testing of our approach in various conditions, we have evaluated Algorithm 1 on synthetic data on which we can control important parameters and guarantee structural or distributional properties.

Different analysis have been conducted. We have measured the impact of the probabilistic models and of the thresholds involved in the validity criteria on the precision of the results and on the pruning ratio. The pruning ratio is the ratio of mappings that are pruned by adding

in line 8 $Implicant(m)$ (or $Implicant_c(m)$) to the set M_{NVal} of unvalid mappings without estimating their probabilities.

We have also measured the qualitative and quantitative impact of the choice of a classifier. Automatic classification is at the core of the estimation of $\widehat{P_u}$ and $\widehat{P_c}$ with the computation of $\widehat{Ext}(C_i \cap D_j, \mathcal{O})$ (see Theorem 2). For evaluating the quality of our results, we use the standard criteria of precision and recall [35]. Recall is the ratio of returned results that are expected w.r.t. all expected results. Precision is the ratio of returned results that are expected w.r.t. all returned results.

Finally, we have tested the robustness of Algorithm 1 to noisy data.

We first describe the principles and the process of the data generator on which we have conducted the different experiments. Then we describe the experimental protocol that we have followed. Finally, we summarize the main experimental results that we have obtained.

5.1 Synthetic data generation

Synthetic data generation is divided into three steps : generation of taxonomies with fixed sizes, generation of the expected mappings to discover, and population of each class by generating a fixed number of instances and associated description.

Generation of taxonomies

Given constraints on number of classes n_1 and n_2, we generate the structure of the two respective taxonomies $\mathcal{T}_1$ and $\mathcal{T}_2$ as a forest of general trees (unconstrained in-degrees) by using a Boltzmann sampler for unlabelled trees described in [14]. We use a reject method to get random forests with n_1 and n_2 nodes. This method is simple and efficient while guaranteeing an uniform distribution among the trees with the same number of nodes. Then, we label each node by a distinct class name.

In our experiments, we set $n_1 = n_2$ so the two taxonomies have the same size, which is the unique parameter of the taxonomies generation.

Mappings generation

We initialize the generation of mappings to be discovered $\mathcal{M}_G$ with a set $\mathcal{M}_S$ of seed mappings, whose size is either fixed or randomly chosen between 1 and the common size of the taxonomies.

Each mapping $m \in \mathcal{M}_S$ is generated by a random choice for the two classes $lhs(m)$ and $rhs(m)$ in $\mathcal{T}_1$ and $\mathcal{T}_2$, or in $\mathcal{T}_2$ and $\mathcal{T}_1$, depending on the mapping direction which is randomly chosen too. We reject mappings which logically entail class inclusions that are not entailed within each taxonomy (i.e., we forbid generated mappings to modify the knowledge of each taxonomy).

The set $\mathcal{M}_G$ of all mappings to discover will then be the set of mappings that can be logically entailed by $\mathcal{M}_S$ and the two taxonomies.

Following [15], the computation of precision and recall will be based on $\mathcal{M}_G$. Let R be the result of Algorithm 1. The recall is the proportion of mappings of $\mathcal{M}_G$ actually returned by Algorithm 1 :

$$Recall = \frac{\mathcal{M}_G \cap R}{\mathcal{M}_G}$$

The precision is the proportion of returned mappings that are actually in $\mathcal{M}_G$:

$$Precision = \frac{\mathcal{M}_G \cap R}{R}$$

Instances and description generation

For this step, we consider the two taxonomies and the mappings between them as a whole DAG of classes. The acyclicity of that graph is guaranteed by the constraints imposed in the production of the set $\mathcal{M}_G$ of generated mappings described above.

We first generate a set of boolean attributes sufficient to associate a minimal intentional description of each class respecting the semantic partial order $\preceq$ conveyed by the above DAG structure. Then, we use this intentional knowledge to generate accordingly the description of the instances with which we populate each class in the taxonomies.

Generation of the intentional description of classes :
We traverse the DAG of classes according to a reverse topological order [6] starting from the most general classes that constitute the level 0, and we iterate the following process for generating the intention of classes as sets of attributes :

- For each class C_i^0 of level 0, we generate a disjoint set of distinct attributes $\mathcal{A}t_i^0$ and we set the intention of C_i^0, denoted $Int(C_i^0)$, to be $\mathcal{A}t_i^0$.
- For each class C_i^j of level j (according to the reverse topogical order), we generate a set $\mathcal{A}t_i^j$ of novel attributes (disjoint from the set of existing attributes) with a size fixed to the out degree of C_i^j in the DAG of classes, and we set $Int(C_i^j)$ to be $\mathcal{A}t_i^j \cup \bigcup Int(C_{i_k}^{j-1})$, where the $C_{i_k}^{j-1}$ are the successors of C_i^j in the DAG.

Population of classes :
Let $\{At_1, \ldots, At_m\}$ be the set of attributes generated at the previous step. We populate each class with n_P instances, and we associate to them descriptions that respect the corresponding intentional description, as follows : For each class C, each of its instances is described by a boolean vector $[a_1, \ldots, a_m]$ obtained by :

- setting to 1 each a_i such that the corresponding attribute At_i is in the intention of the class C,
- randomly setting the other values a_j to 0 or 1.

This way, by construction, all the instances in the extension of a class have in common that all the attributes in $Int(C)$ are present in their description.

In Section 5.3 we will use an *oracle classifier* which classifies an instance i in the class C iff all the attributes of the intention $Int(C)$ of the class C are present in the description of i.

The results of data generation can be summarized into a table T_{data} with $m + n_C$ columns where m is the number of generated attributes and n_C is the number of generated classes, and each tuple $[a_1, \ldots a_m, c_1, \ldots, c_{n_c}]$ concatenates the description $[a_1, \ldots a_m]$ of an instance in terms of attributes, and its categorization $[c_1, \ldots, c_{n_c}]$ with respect to the classes : for each $i \in [1..n_C]$ c_i is equal to 1 if $i \in Ext(C)$ and to 0 if it is not the case.

Connection with Armstrong relations

We conclude this section on data generation by outlining a connection between the table T_{data} and the construction of *Armstrong relations* [19]. An Armstrong relation for a set of implicational dependencies is a relation that satisfies each dependency implied by the set and does not satisfy any dependency that is not implied by it. The tuples in the table T_{data} can be seen as a sample of the tuples in the Amstrong relation for the set $\{C \leftrightarrow At_1, \ldots, At_k\}$ of implicational dependencies existing between each class C and the attributes $\{At_1, \ldots, At_k\}$. It is known ([19]) that the number of tuples in a minimal Armstrong relation maybe be exponential in the number of columns. The interesting point in our data generation is that we control the number of tuples in T_{data} to be polynomial in function of the number of classes, and we guarantee by construction that the expected implications between classes are verified. We

cannot guarantee that T_{data} does not satisfy some unexpected implications between classes as an effect of the random choice of free attributes when generating the instance descriptions. However, we have shown experimentally (on taxonomies of size 30 with a population of 200 instances per class) that the probabilities (inferred by the oracle classifier) of the unexpected implications between classes are very low.

5.2 Experimental protocol

We first explain the goals of the successive experiments that we have performed.

The first goal is to analyze the impact on the precision of the thresholds S_c, S_u involved in the different validity criteria, with the purpose to fix appropriate thresholds for the next experiments.

The second goal is to analyze the impact of the probabilities $\widehat{P_c}$ and $\widehat{P_u}$ on the pruning ratio of the algorithm. The purpose is to determine among the three validity criteria defined in Section 4 the one offering the best performances.

The third goal is to analyse and compare the impact both on precision/recall and on total running time of three real classifiers (Naive Bayes, C4.5 and SVM) for estimating the probabilities. The purpose is to determine the classifier offering the best tradeoff between quality of results and running time. Note that we do not take the learning time of classifiers into account because we consider that this task can be precomputed for each taxonomy.

Finally, the last goal is to analyse the robustness of the approach to noisy data.

For all the experiments presented in this section, each point is obtained by averaging the results of 100 runs. For each run, a new synthetic dataset is generated with the appropriate parameters. Note that in our experiments we generate taxonomies with few dozens of classes. The number of random taxonomies of such sizes can be counted in billions. Thus, averaging over 100 runs for a point does not prevent from local variations, leading to curves that are not smooth.

Our algorithm is written in Java and compiled using Sun Java version 1.6. We run all the tests on a quad-core Intel Q6700 Xeon at 2.66 GHz with 4 GB of memory. The OS is Ubuntu Linux 8.10. For all the experiments measuring run times, only one instance of our program and the OS are running on the machine, to avoid memory contention effects with other programs that would affect the results.

5.3 Experimental results

5.3.1 Impact of thresholds on precision

We compare the influence of the thresholds S_c and S_u associated to probabilities $\widehat{P_c}$ and $\widehat{P_u}$ on the quality of the results returned by Algorithm 1. For doing so, we run the algorithm with the validity criterion : $\widehat{P_c} \geq S_c$ and $\widehat{P_u} \geq S_u$.

In this experiment, the computation of probabilities is performed using the oracle classifier. The parameters in the synthethic generator are defined such that $|\mathcal{M}(\mathcal{T}_1, \mathcal{T}_2)| = 320$. We set the number of seed mappings $|\mathcal{M}_S| = 4$. Note that by logical entailment the total number $|\mathcal{M}_G|$ of mappings to be discover may be much greater. For each couple of threshold $(S_c, S_u) \in [0.78 : 0.995]^2$, we compute the precision and the recall of the results of Algorithm 1. We observed that the recall remains constant at 1.0 independently of values of S_c and S_u. This is because thanks to the oracle classifier, estimated probabilities for the mappings of $\mathcal{M}_G$ are very close to 1, and superior to all experimented thresholds, leading to a perfect recall. Thus, we only show the results for precision in Figure 3.

The figure shows the contours of the different precision levels, from a precision of 0.93 to a precision of 0.99. From the shape of these contours, it is clear that both $\widehat{P_c}$ and $\widehat{P_u}$ have an

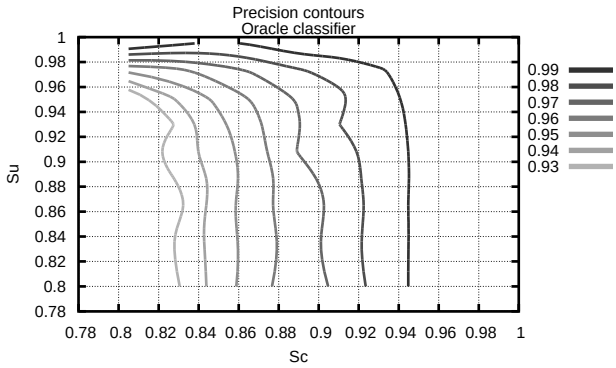


FIG. 3 – Precision w.r.t. S_c and S_u thresholds

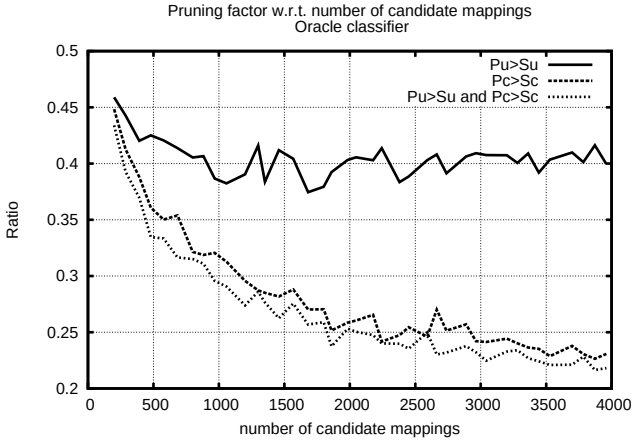


FIG. 4 – Pruning factor for validity computation with $\widehat{P_c}$, $\widehat{P_u}$ and $\widehat{P_c}$ *and* $\widehat{P_u}$

influence on precision. As the relation $\widehat{P_u} \geq \widehat{P_c}$ holds (Proposition 2), under the diagonal $\widehat{P_u}$ has no influence on precision.

The probability $\widehat{P_c}$ is more discriminant than $\widehat{P_u}$. The figure shows that $\widehat{P_c}$ influences the precision for a large range of values of the threshold S_c, while $\widehat{P_u}$ only has an influence for very high values of S_u. We have observed the estimated probabilities for different mappings, and found that there is an important gap in the values of $\widehat{P_c}$ between valid and invalid mappings. This gap is much smaller for $\widehat{P_u}$. $\widehat{P_u}$ giving higher probability values to invalid mappings, this explains why it can only have an influence on precision at very high S_u values.

Based on the curve of Figure 3, we fix the thresholds at $(S_c = 0.83, S_u = 0.96)$ for experiments where the classifier used to estimate the probabilities is the oracle. This gives a good precision of 0.95, and maps to a region where $\widehat{P_u}$ has an influence on the quality of results.

For the experiments in which a real classifier is used to estimate the probabilities, we fix the thresholds at $(S_c = 0.85, S_u = 0.90)$ to be tolerant to classification errors.

5.3.2 Impact of probabilities on pruning ratio

We now study the impact of the probabilistic criteria defined in Section 4 for testing the validity of a mapping on the pruning ratio performed by Algorithm 1.

Figure 4 shows the ratio of pruning made by the algorithm using the different validity criteria, w.r.t. a naive approach not doing any pruning.

The validity computation using only $\widehat{P_u}$ is the one that prunes the least mappings, computing

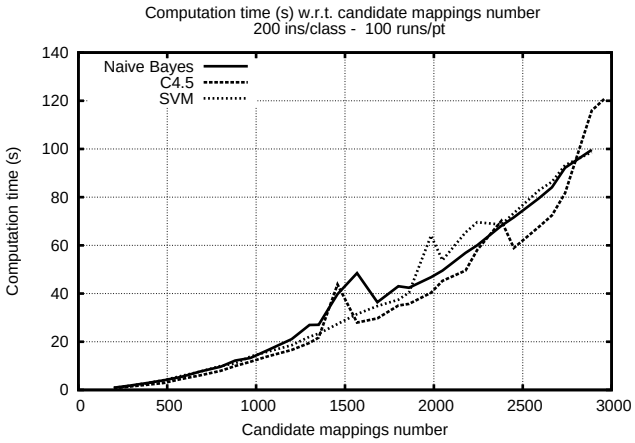


FIG. 5 – Computation time (s) for different classifiers

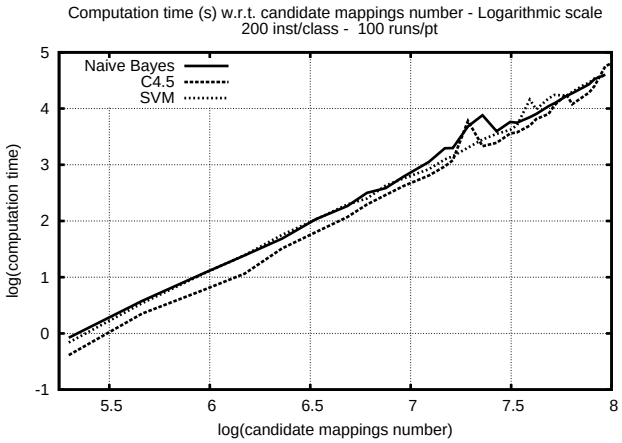


FIG. 6 – Computation time in log scale for different classifiers

probabilities for about 40% of all mappings of $\mathcal{M}(\mathcal{T}_i, \mathcal{T}_j)$. Both $\widehat{P_c}$ and $\widehat{P_c}$ *and* $\widehat{P_u}$ does more prunings and obtain a significant reduction of the search space. Combining $\widehat{P_u}$ and $\widehat{P_c}$ obtains slightly better results than using $\widehat{P_c}$ alone, so for the remainder of this experiments section, we use $\widehat{P_c} \geq S_c$ and $\widehat{P_u} \geq S_u$ as validity criterion. It allows to compute the probabilities for only 20% of the mappings of $\mathcal{M}(\mathcal{T}_i, \mathcal{T}_j)$, when the number of candidate mappings is high.

5.3.3 Impact of the classifiers

In this subsection, we replace the oracle classifier with a real classifier. We compare the results given by three well-known classifiers : Naive Bayes [29], C4.5 [30] and SVM [20]. We use the Weka implementation of these classifiers and have interfaced it with our code.

The comparisons of running times are shown in Figure 5 and in log scale in Figure 6.

A first conclusion is that the running times are polynomial in the number of mappings, and are very similar, with Naive Bayes being slightly slower than C4.5 and SVM.

Comparisons for precision and recall are shown in respectively Figure 7 and Figure 8. Whatever the classifier, precision is not impacted by the number of candidate mappings, i.e. the size of search space. This is also the case for the recall, except when the classifier is Naive Bayes.

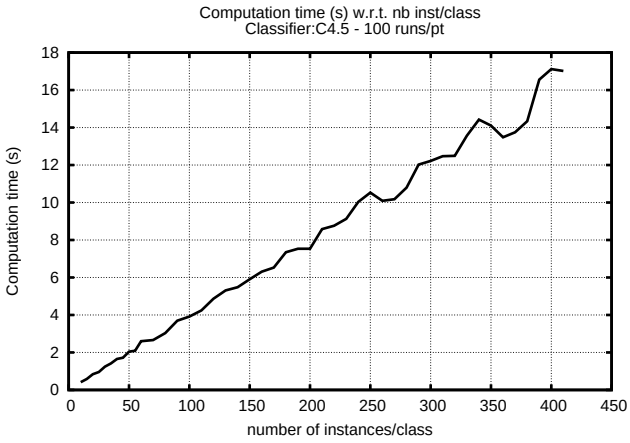


FIG. 9 – C4.5 - Computation time (s) - impact of number of inst/class

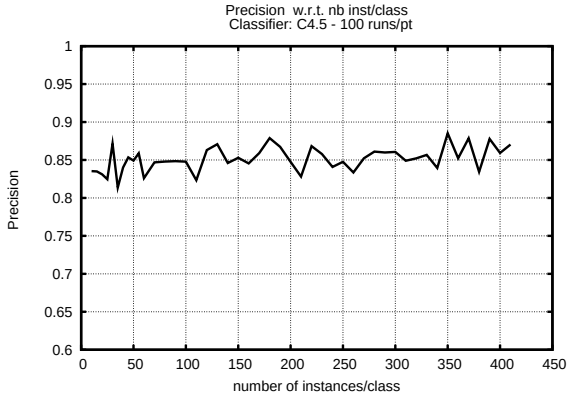


FIG. 10 – C4.5 - Precision - impact of number of inst/class

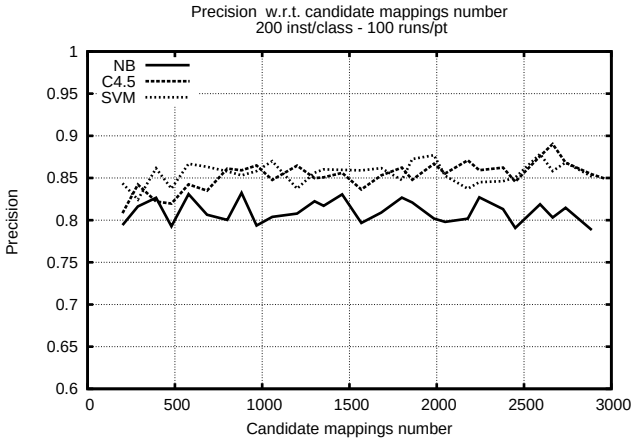


FIG. 7 – Precision for different classifiers

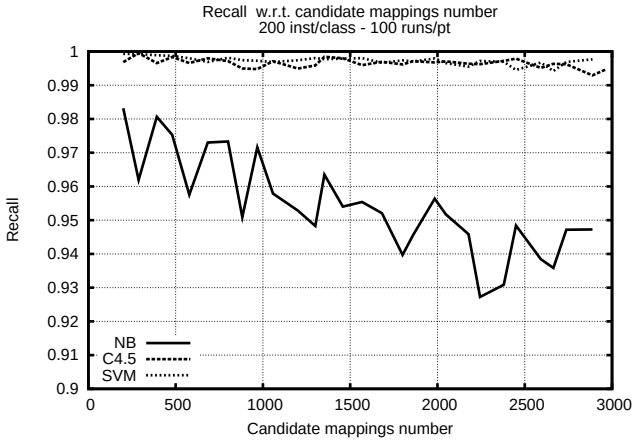


FIG. 8 – Recall for different classifiers

Naive Bayes has both the worst recall and the worst precision, the choice is thus between C4.5 and SVM. They seem to have similar results. However, the learning time of SVM (not shown here) is much longer than the learning time of C4.5. We thus choose C4.5 for further experiments, and analyse the impact of the number of instances per class on the classification performance of Algorithm 1 with C4.5.

We vary the number of instances per class n_P between 10 and 450. The results for computation time, precision and recall are shown in Figures 9, 10 and 11.

In this experiment, the number of classes and of mapping is constant, hence the number

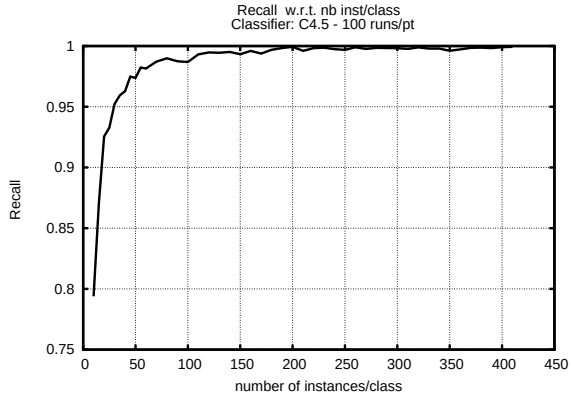


FIG. 11 – C4.5 - Recall - impact of number of inst/class

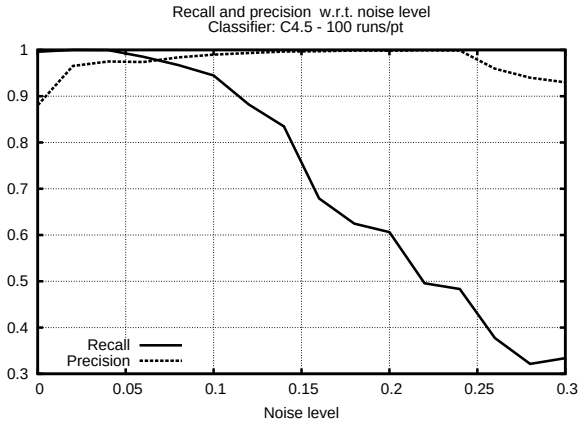


FIG. 12 – C4.5 - Precision and recall - impact of noise level

of classifications to perform is linear in the number of instances. The C4.5 algorithm takes linear time in the number of instances. As expected, this is also the case for Algorithm 1, as shown by Figure 9. Increasing the number of instances per class only increases slightly precision, whereas it strongly improves recall. The most important point to note is that excellent values of precision and recall are obtained with as few as 50 instances per class, as expected, with a use of a bayesian approach of statistics.

5.3.4 Robustness to noisy data

In order to test the robustness to noise of our algorithm, we define a new parameter θ corresponding to the quantity of noise to inject in the synthetic data. Each dataset produced by the synthetic data generator goes through a step of noise application, where each boolean corresponding to the value of an attribute for an instance can be reversed with a probability θ. The new dataset is then processed as usual by Algorithm 1.

The variations of precision and recall for values of $\theta \in [0; 0.3]$ are show in Figure 12.

The figure shows that recall gracefully degrades when noise increases. At 10% noise, the recall is nearly unaffected, at a value of 0.95. Values of noise superior to 15% have a more significant impact and lead to poor recall.

Precision, however, exhibits a different behavior. It first *increases* with noise, before abruptly decreasing for more than 24% of noise.

In order to understand this phenomenon, we have investigated in details the classifier results and the values of probabilities given to mappings. We found that for 0% noise, there are invalid mappings that are incorrectly given too high probabilities, and that appear as valid. This

explains the non-perfect 0.88 precision value. The probability values for these mappings are close to the threshold. Increasing noise makes the classifiers more selective, and tends to decrease the values of all probabilities. So the probabilities of these invalid mappings go below the threshold for a moderate amount of noise, whereas the probabilities of valid mappings remain above the threshold. Thus the precision increases.

6 Related work and conclusion

As outlined in the introduction, semantic mappings are the glue for data integration systems. A wide range of methods of schema/ontology matching have been developed both in the database and the semantic web communities [17]. One of the principles widely exploited is terminological comparison of the labels of classes with string-based similarities or lexicon-based similarities (like WordNet) (e.g., TaxoMap [22], H-MATCH [5]) . Another widely used principle is structure comparison between labeled graphs representing ontologies (e.g., OLA [18]). In fact, most of the existing matchers combine these two approaches in different ways (e.g., COMA++ [2] and COMA [10], Cupid [27], H-MATCH [5]). Other approaches have been investigated using machine learning techniques using a corpus of schema matches (e.g., [26]), or a corpus of labelled instances (e.g., LSD [11], SemInt [25], GLUE [12], FCA-merge [33]).

It is standard practice for ontology and schema matchers to associate numbers with the candidate mappings they propose. However, those numbers do not have a probabilistic meaning and are just used for ranking.

In contrast, our approach promotes a probabilistic semantics for mappings and provides a method to compute mapping probabilities based on the descriptions of instances categorized in each ontology. It is important to note that even if we use similar classification techniques as [12], we use them for computing true probabilities and not similarity coefficients. In addition, we follow a Bayesian approach for estimating joint probabilities which is more robust to a small number of instances.

The most distinguishing feature of our approach is that it bridges the gap between logic and probabilities by providing probabilistic models that are consistent with the logical semantics underlying ontology languages. Therefore, our approach generalizes existing works based on algebraic or logical representation of mappings as a basis for reasoning (e.g., S-Match [21], Clio [28]).

The thorough experiments that we have conducted on controlled synthetic data have shown the feasibility and the scalability of our approach. We have experimentally demonstrated that the time complexity of our algorithm is polynomial in the size of the taxonomies to align and linear in the number of instances. We have also shown that our algorithm has a good robustness w.r.t. noisy data.

The mappings that are returned by our algorithm can be exploited for mapping validation by probabilistic reasoning in the line of what is proposed in [4].

More generally, our approach is complementary of the recent work that has been flourishing on probabilistic databases [3, 7]. In particular, it fits into the general framework set in [13] for handling uncertainty in data integration, for which it provides an effective way for computing mapping probabilities.

As a future work, we plan to integrate our method in the SomeWhere peer-to-peer infrastructure [1] in order to discover automatically mappings between a peer joining the network and its acquainted peers. We will conduct an experiment on the deployment of SomeWhere enriched with automatic mapping discovery for peer-to-peer data sharing of musical files, in which each user will manually categorize his music files according to his own taxonomy. Our method will exploit the metadata that can be extracted from several formats of audio files.

For example, MP3 files embed descriptive tags complying to the ID3 [24] standard for metadata. These tags specify features such as the title, author, genre of the song encoded. Those attribute-value metadata can be extracted automatically from MP3 files and can be converted by adequate preprocessing into a set of boolean attributes for a direct application of our mapping discovery method.

Références

[1] P. Adjiman, P. Chatalic, F. Goasdoué, M.-C. Rousset, and L. Simon. Distributed reasoning in a peer-to-peer setting : Application to the semantic web. *J. Artif. Intell. Res. (JAIR)*, 25 :269–314, 2006.

[2] D. Aumueller, H. H. Do, S. Massmann, and E. Rahm. Schema and ontology matching with coma++. In *SIGMOD '05 : Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, pages 906–908, New York, NY, USA, 2005. ACM.

[3] O. Benjelloun, A. D. Sarma, A. Y. Halevy, and J. Widom. Uldbs : Databases with uncertainty and lineage. In *VLDB*, pages 953–964, 2006.

[4] S. Castano, A. Ferrara, D. Lorusso, T. H. Näth, and R. Möller. Mapping validation by probabilistic reasoning. In *Proc. 5th European Semantic Web Conference (ESWC 2008)*, 2008.

[5] S. Castano, A. Ferrara, and S. Montanelli. H-match : an algorithm for dynamically matching ontologies in peer-based systems. In *SWDB*, pages 231–250, 2003.

[6] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms, Second Edition*. The MIT Press, September 2001.

[7] N. N. Dalvi and D. Suciu. Answering queries from statistics and probabilistic views. In *VLDB*, pages 805–816, 2005.

[8] M. Dean and G. Schreiber. OWL web ontology language reference. W3C recommendation, W3C, February 2004.

[9] M. H. Degroot. *Optimal Statistical Decisions (Wiley Classics Library)*. Wiley-Interscience, April 2004.

[10] H. H. Do and E. Rahm. Coma - a system for flexible combination of schema matching approaches. In *VLDB*, 2002.

[11] A. Doan, P. Domingos, and A. Y. Levy. Learning mappings between data schemas. In *Proceedings of the AAAI-2000 Workshop on Learning Statistical Models from Relational Data*, 2000.

[12] A. Doan, J. Madhavan, P. Domingos, and A. Y. Halevy. Learning to map between ontologies on the semantic web. In *WWW*, pages 662–673, 2002.

[13] X. L. Dong, A. Y. Halevy, and C. Yu. Data integration with uncertainty. In *VLDB*, pages 687–698, 2007.

[14] P. Duchon, P. Flajolet, G. Louchard, and G. Schaeffer. Boltzmann samplers for the random generation of combinatorial structures. *Comb. Probab. Comput.*, 13(4-5) :577–625, 2004.

[15] J. Euzenat. Semantic precision and recall for ontology alignment evaluation. In *IJCAI*, pages 348–353, 2007.

[16] J. Euzenat. Ontology alignment evaluation initiative, July 2008. `http ://oaei.ontologymatching.org/`.

[17] J. Euzenat and P. Shvaiko. *Ontology matching*. Springer-Verlag, Heidelberg (DE), 2007.

[18] J. Euzenat and P. Valtchev. Similarity-based ontology alignment in owl-lite. In *ECAI*, pages 333–337, 2004.

[19] R. Fagin. Horn clauses and database dependencies. *J. ACM*, 29(4) :952–985, 1982.

[20] G. W. Flake and S. Lawrence. Efficient svm regression training with smo. *Machine Learning*, 46(1-3) :271–290, 2002.

[21] F. Giunchiglia, P. Shvaiko, and M. Yatskevich. S-match : an algorithm and an implementation of semantic matching. In *ESWS*, pages 61–75, 2004.

[22] F. Hamdi, H. Zargayouna, B. Safar, and C. Reynaud. TaxoMap in the OAEI 2008 alignment contest . In *Ontology Alignment Evaluation Initiative (OAEI) 2008 Campaign - Int. Workshop on Ontology Matching*, 2008.

[23] P. Hayes, editor. *RDF Semantics*. W3C Recommendation. W3C, February 2004.

[24] Id3 official site. `http ://www.id3.org/`.

[25] W.-S. Li and C. Clifton. Semint : a tool for identifying attribute correspondences in heterogeneous databases using neural networks. *Data Knowl. Eng.*, 33(1) :49–84, 2000.

[26] J. Madhavan, P. A. Bernstein, A. Doan, and A. Halevy. Corpus-based schema matching. *ICDE*, pages 57–68, 2005.

[27] J. Madhavan, P. A. Bernstein, and E. Rahm. Generic schema matching with cupid. In *The VLDB Journal*, pages 49–58, 2001.

[28] R. J. Miller, L. M. Haas, and M. A. Hernández. Schema mapping as query discovery. In A. E. Abbadi, M. L. Brodie, S. Chakravarthy, U. Dayal, N. Kamel, G. Schlageter, and K.-Y. Whang, editors, *VLDB*, pages 77–88. Morgan Kaufmann, 2000.

[29] T. Mitchell. *Machine Learning*. McGraw-Hill Education (ISE Editions), October 1997.

[30] R. J. Quinlan. *C4.5 : Programs for Machine Learning (Morgan Kaufmann Series in Machine Learning)*. Morgan Kaufmann, January 1993.

[31] E. Rahm and P. A. Bernstein. A survey of approaches to automatic schema matching. *The VLDB Journal*, 10(4) :334–350, 2001.

[32] P. Shvaiko and J. Euzenat. A survey of schema-based matching approaches. *J. Data Semantics*, pages 146–171, 2005.

[33] G. Stumme and A. Maedche. FCA-MERGE : Bottom-Up Merging of Ontologies. In *IJCAI*, pages 225–234, 2001.

[34] R. Tournaire and M.-C. Rousset. Découverte automatique de correspondances entre taxonomies - internal report (in french), 2008. `http ://membres-liglab.fr/tournaire/Remi.Tournaire/irap08.pdf`.

[35] C. J. Van Rijsbergen. *Information retrieval*. Butterworths, London, Boston, 1975.

[36] I. H. Witten and E. Frank. *Data Mining : Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, 2nd edition, 2005.