Discovery of Probabilistic Mappings between Taxonomies: Principles and Experiments

Rémi Tournaire[1], Jean-Marc Petit[2] Marie-Christine Rousset[1], and Alexandre Termier[1]

[1] University of Grenoble, Laboratory of Informatics of Grenoble UMR 5217,
681, rue de la Passerelle, 38402 St-Martin d'Hères Cedex, France
Remi.Tournaire@imag.fr
[2] INSA Lyon, LIRIS UMR 5205, 69621 Villeurbanne Cedex, France

Abstract. In this paper, we investigate a principled approach for defining and discovering *probabilistic mappings* between two taxonomies. First, we compare two ways of modeling probabilistic mappings which are compatible with the logical constraints declared in each taxonomy. Then we describe a *generate and test* algorithm which minimizes the number of calls to the probability estimator for determining those mappings whose probability exceeds a certain threshold. Finally, we provide an experimental analysis of this approach.

1 Introduction

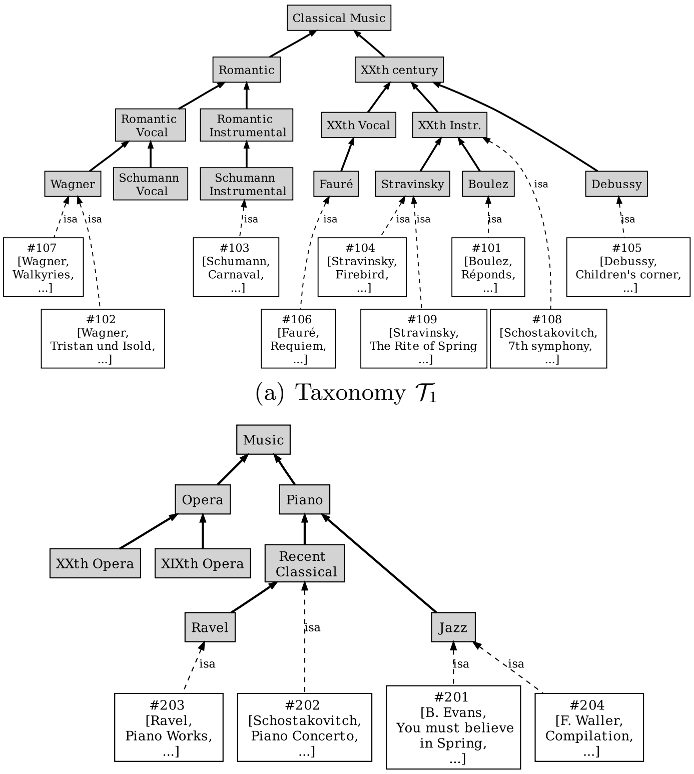
The decentralized nature of the development of Web data management systems makes inevitable the independent construction of a large amount of personalized taxonomies used for annotating data and resources at the Web scale. Taxonomies are hierarchical structures appropriate to data categorization and semantic annotation of resources. They play a prominent role in the Semantic Web since they are central components of OWL [1] or RDF(S) [2] 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.

In this work we focus on discovering mappings between populated taxonomies like Web Directories or folksonomies used for example for organizing musical songs. Web Directories like the Yahoo! Directory[3] and the Google directory[4] (see Figure 2) can be considered as taxonomies because they are hierarchical structures categories organizing web pages. Folksonomies like those represented in Figure 1 are small taxonomies created by final users and populated with musical files, for example.

[3] dir.yahoo.com

[4] dir.google.com


Classical Music
Romantic
XXth century
Romantic Vocal
Romantic Instrumental
XXth Vocal
XXth Instr.
Wagner
Schumann Vocal
Schumann Instrumental
Fauré
Stravinsky
Boulez
Debussy
isa
#107 [Wagner, Walkyries, ...]
#102 [Wagner, Tristan und Isold, ...]
#103 [Schumann, Carnaval, ...]
#106 [Fauré, Requiem, ...]
#104 [Stravinsky, Firebird, ...]
#109 [Stravinsky, The Rite of Spring ...]
#101 [Boulez, Réponds, ...]
#108 [Schostakovitch, 7th symphony, ...]
#105 [Debussy, Children's corner, ...]
Music
Opera
Piano
XXth Opera
XIXth Opera
Recent Classical
Ravel
Jazz
#203 [Ravel, Piano Works, ...]
#202 [Schostakovitch, Piano Concerto, ...]
#201 [B. Evans, You must believe in Spring, ...]
#204 [F. Waller, Compilation, ...]

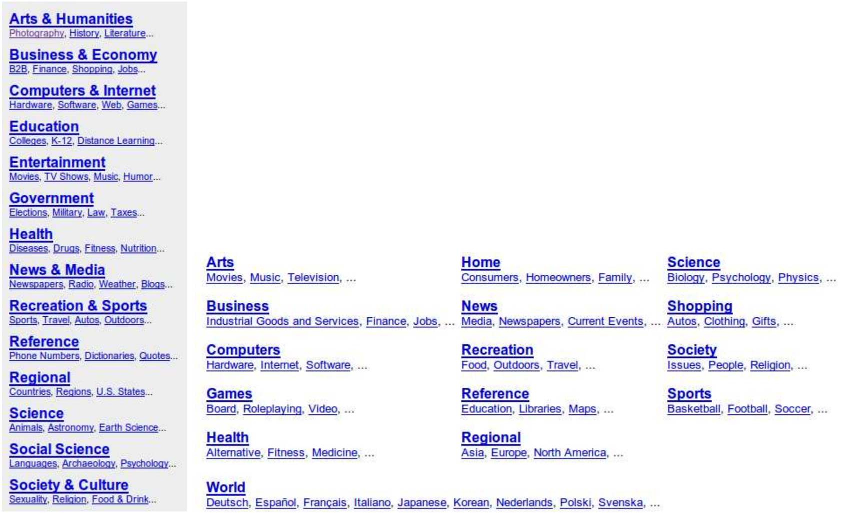
(a) Taxonomy $\mathcal{T}_1$

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

Fig. 1. 2 Taxonomies and associated instances

Providing mappings between two taxonomies makes possible an exchange of documents (instances) by querying one taxonomy with the vocabulary of the other one. For instance, mappings between Yahoo! and Google directories allow to query both of them for pages about “Autos” while covering both the pages indexed by Yahoo! and those indexed by Google. With folksonomies of Figure 1, a local query of songs about “XXth Vocal” may be completed by songs populating the distant class *XXth Opera* thanks to a provided mapping representing the fact that *XXth Opera* is included in *XXth Vocal*. We claim that such an application of joint querying requires a grounded and semantic way for defining the mappings and measuring their degree of confidence.

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 [3, 4] for surveys).



Arts & Humanities
Photography, History, Literature...

Business & Economy
B2B, Finance, Shopping, Jobs...

Computers & Internet
Hardware, Software, Web, Games...

Education
Colleges, K-12, Distance Learning...

Entertainment
Movies, TV Shows, Music, Humor...

Government
Elections, Military, Law, Taxes...

Health
Diseases, Drugs, Fitness, Nutrition...

News & Media
Newspapers, Radio, Weather, Blogs...

Recreation & Sports
Sports, Travel, Autos, Outdoors...

Reference
Phone Numbers, Dictionaries, Quotes...

Regional
Countries, Regions, U.S. States...

Science
Animals, Astronomy, Earth Science...

Social Science
Languages, Archaeology, Psychology...

Society & Culture
Sexuality, Religion, Food & Drink...

(a) Sample directory from Yahoo!

Arts
Movies, Music, Television, ...

Business
Industrial Goods and Services, Finance, Jobs, ...

Computers
Hardware, Internet, Software, ...

Games
Board, Roleplaying, Video, ...

Health
Alternative, Fitness, Medicine, ...

Home
Consumers, Homeowners, Family, ...

News
Media, Newspapers, Current Events, ...

Recreation
Food, Outdoors, Travel, ...

Reference
Education, Libraries, Maps, ...

Regional
Asia, Europe, North America, ...

Science
Biology, Psychology, Physics, ...

Shopping
Autos, Clothing, Gifts, ...

Society
Issues, People, Religion, ...

Sports
Basketball, Football, Soccer, ...

World
Deutsch, Español, Français, Italiano, Japanese, Korean, Nederlands, Polski, Svenska, ...

(b) Sample directory from Google

Fig. 2. Top levels of Yahoo! and Google directories

Particular focus and applications of the proposed approach

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. [5–8]).

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 ([9–11]). Those coefficients are the basis for yearly international comparative evaluation campaigns [12, 13]. 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., [14]). 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 subsumption 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 article, we focus on *inclusion mappings*. Inclusion mappings are of primary importance for ontology alignment applications, but they have received only little attention until now. In addition, inclusion mappings with a probabilistic semantics are useful and easily interpretable by human or automatic systems, because they combine several theoretical and practical advantages:

1. Inclusion mappings are more likely to happen than equivalences (theoretically at least twice), and so our method discovers more information than methods based on equivalences.
2. Inclusion mappings are logically weaker than equivalences but have a clear logical semantic that is directly useful for integration and query rewriting settings. For example, the inclusion mapping

   $$\texttt{Google/Software/Games} \sqsubseteq \texttt{Yahoo/Entertainment}$$

   allows to make a query about `Yahoo/Entertainment` (or any of its superclasses) and leads to answer this query by considering that all pages in the category
   `Google/Software/Games` should be in the answer. In this way, an alignment between Yahoo! and Google directories constituted by inclusion mappings may enable an access to all referenced pages of both Yahoo! and Google with a single query. A large amount of relevant mappings (like the above mapping) could not have been discovered by methods that return only equivalences.
3. The probabilities associated to mappings can be used by others systems that require probabilities, for example in order to know what are the most reliable knowledge. Properly handling uncertainty is a key point for the Semantic Web due to the heterogeneous and unreliable nature of available data. In particular, probabilistic inclusion mappings directly fit the framework of probabilistic databases [16] and probabilistic reasoning [17]. These works contribute to probabilistic query answering which constitutes a major way to exchange data between heterogeneous schemas and ontologies.
4. In a probabilistic query answering setting, users can obtain probabilistic answers to their queries, thanks to a probabilistic reasoning using probabilistic mappings. From an user point-of-view, a probability has a well-understood meaning when standing for confidence: a probability of 0.5 means there is an even chance the mapping would be correct.
5. Probability theory is quite simple, general, popular and grounded for dealing with uncertainty, and the generality and reusability is a key point in computer science.
6. Finally, from a theoretical point of view in the ontology matching domain, there is a need to better understand the fundations of the uncertainty in data and schema integration ([15]).

Main points and organization of this article

In this paper, we propose an approach to discover automatically *probabilistic inclusion mappings* between classes of taxonomies, in order to query multiple folksonomies or Web Directories in a way that is grounded, robust and easily interpretable.

First, we investigate and compare two instance-based 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 *monotony* 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 monotony 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 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 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) [2] 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 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 taxonomy 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 $XXth\ Vocal_1$.

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*
- $\mathcal{T}_j \models rhs(m) \sqsubseteq rhs(m')$

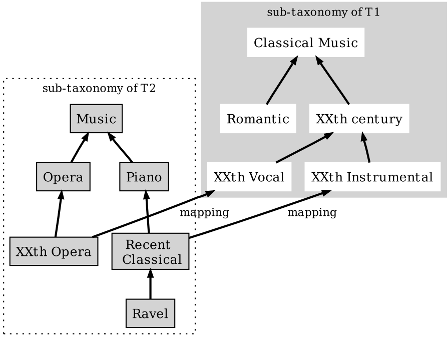


Fig. 3. 2 mappings between $\mathcal{T}_1$ and $\mathcal{T}_2$

For example, two mappings between taxonomies $\mathcal{T}_1$ and $\mathcal{T}_2$ of Figure 1 are illustrated in Figure 3:

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

Theorem 1 (Property of monotony). *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 [18] 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 $Pr(lhs(m) \cap rhs(m))$. These 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 [19], we model these (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, [19]), 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 [20] for a survey) and have been made available through open source platforms like Weka [21]. 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 [5]:

- $\mathcal{C}$ 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 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 [5]). 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 of two random sets respectively for a uniform distribution of instances in the sample set.

4 Discovery process of probabilistic mappings

Given two taxonomies $\mathcal{T}_i$ and $\mathcal{T}_j$ (and their associated instances), let $\mathcal{M}(\mathcal{T}_i, \mathcal{T}_j)$ be the set of all mappings from $\mathcal{T}_i$ to $\mathcal{T}_j$ (i.e., of the form $C_i \sqsubseteq D_j$). The ProbaMap algorithm determines all mappings m of $\mathcal{M}(\mathcal{T}_i, \mathcal{T}_j)$ that verify two probabilistic-based criterion of $\widehat{P}_u(m) \geq S_u$ and $\widehat{P}_c(m) \geq S_c$, where S_u and S_c are two thresholds in $[0, 1]$.

Candidate mapping generation The principle of ProbaMap algorithm is to generate mappings from the two sets of classes in the two taxonomies ordered according to a *topological sort* [22]. Namely, (see the nested loops (Line 4) in Algorithm 1) it generates all the mappings $C_i \sqsubseteq D_j$ by enumerating the classes C_i of $\mathcal{T}_i$ following a *reverse* topological order and the classes D_j of $\mathcal{T}_j$ following a *direct* topological order. The following proposition is a corollary of Proposition 1.

Proposition 3. *Let $\mathcal{T}_i$ and $\mathcal{T}_j$ two taxonomies.*
Let $ReverseTopo(\mathcal{T}_i)$ be the sequence of classes of $\mathcal{T}_i$ resulting from a reverse topological sort of $\mathcal{T}_i$. Let $Topo(\mathcal{T}_j)$ be the sequence of classes of $\mathcal{T}_j$ resulting from a topological sort of $\mathcal{T}_i$. Let $m : C_i \sqsubseteq D_j$ and $m' : C'_i \sqsubseteq D'_j$ two mappings from $\mathcal{T}_i$ to $\mathcal{T}_j$. If m' is an implicant of m (i.e., $m' \preceq m$), then C_i is before C'_i in $ReverseTopo(\mathcal{T}_i)$ or $C_i = C'_i$ and D_j is before D'_j in $Topo(\mathcal{T}_j)$.

This proposition ensures that the mappings are generated in an order that respect the logical implication, according to the knowledge of the taxonomies.

Pruning the candidate mappings to test Based on the monotony property of the probability function P_u (Theorem 1), every mapping m' implicant of a mapping m such that $P_u(m) < S_u$ verifies $P_u(m') < S_u$. Therefore, in ProbaMap, we prune the probability estimation of all the implicants of every m such that $\widehat{P_u}(m) < S_u$. We shall use the notation $Implicants(m)$ to denote the set of all mappings that are implicants of m. Similarly, based on the property of weak monotony of the probability function P_c (Theorem 1), when a tested candidate mapping m is such that $\widehat{P_c}(m) < S_c$ we prune the probability estimation of all the implicants of 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 $\mathcal{T}_j$.

The ProbaMap Algorithm The resulting ProbaMap algorithm is described in Algorithm 1, in which

- primitive functions like Implicants and Implicants_c returns the implicants of the mapping in argument and the implicants having the same class at the left-hand side.
- Topo andReverseTopo return the sequences of classes of a taxonomy that respect the logical implication order (respectively in the direct and reverse directions).
- ext_i and ext_j represent the two maps that store the extension of each class of respectively $\mathcal{T}_i$ and $\mathcal{T}_j$, or the extensions obtained by classification when it is enabled.

Algorithm 1 returns mappings directed from $\mathcal{T}_i$ to $\mathcal{T}_j$. In order to obtain *all* valid mappings, it must be applied again by swapping its inputs $\mathcal{T}_i$ and $\mathcal{T}_j$.

Algorithm 1 ProbaMap

```
Require: Taxonomies (DAG) $\mathcal{T}_i, \mathcal{T}_j$, thresholds $S_c$, $S_i$, Instances maps $ext_i, ext_j$
Ensure: return $\{m \in \mathcal{M}(\mathcal{T}_i, \mathcal{T}_j)$ such that $\widehat{P_i}(m) \geq S_i$ and $\widehat{P_c}(m) \geq S_c\}$
 1: $M_{Val} \leftarrow \emptyset$
 2: $M_{NVal} \leftarrow \emptyset$
 3: for all $C_i \in$ ReverseTopo($\mathcal{T}_i$) do
 4:   for all $D_j \in$ Topo($\mathcal{T}_j$) do
 5:     let $m = C_i \sqsubseteq D_j$
 6:     if $m \notin M_{NVal}$ then
 7:       if $\widehat{P_i}(m, ext_i, ext_j, \mathcal{T}_i, \mathcal{T}_j) \geq S_i$ then
 8:         if $\widehat{P_c}(m, ext_i, ext_j, \mathcal{T}_i, \mathcal{T}_j) \geq S_c$ then
 9:           $M_{Val} \leftarrow M_{Val} \cup \{m\}$
10:         else
11:           $M_{NVal} \leftarrow M_{NVal} \cup$ Implicants_c($m, \mathcal{T}_j$) {Pruning using the weak monotony}
12:         end if
13:       else
14:         $M_{NVal} \leftarrow M_{NVal} \cup$ Implicants($m, \mathcal{T}_i, \mathcal{T}_j$) {Pruning using the strong monotony}
15:       end if
16:     end if
17:   end for
18: end for
19: return $M_{Val}$
```

The discovered valid mappings are stored in the set M_{NVal}. Mappings that are pruned are stored in the set M_{NVal}. The nested loops in Line 4 in Algorithm

1 generate all the mappings $C_i \sqsubseteq D_j$ from the most general to the most specific mapping. Based on this, Proposition 3 shows that this algorithm maximizes the number of pruning.

When a mapping is generated and if it is not already pruned, it is firstly tested with regard to $P_i(m) \geq S_i$ in Line 7, then if it is positive, it is tested with $P_c(m) > S_c$ in Line 8. In the case $P_i(m) < S_i$, all the implicants of m are marked as pruned (Line 14), thanks to the strong monotony property of P_i. If $P_i(m) \geq S_i$ but $P_c(m) < S_c$, then the property of weak monotony conducts to prune all the implicants of m with the same left-hand side, in Line 11 and they are added to M_{NVal}. Hence, M_{NVal} set is kept up to date by containing all the pruning mappings from the beginning.

The general structure for the implementation of ProbaMap is pictured in Figure 4.

1. Computation of the transitive logical closure of the two taxonomies and their instances given as input. (in order to obtain quickly the class extensions and to optimize the pruning process)
2. Automatic classification step in order to merge the instances of the two taxonomies. This step has been detailed in section 3.4. It can be disabled by the user.
3. Core of ProbaMap: generation-and-test of candidate mapping, according to the two thresholds as input parameters.

The output is directly the set of mappings from the first taxonomy to the second one for which P_u and P_c both exceed their respective thresholds S_u and S_c. We insist on the fact that, once the parameters are set, the discovery process is *fully automatic*.

As many matching methods provide equivalences as their results, we can add a postprocessing for ProbaMap in order to obtain equivalences logically implied by the inclusion mappings discovered. In this case, the thresholds S_u and S_c should be lowered with respect to the discovery of pure implications, because equivalences are stronger relations than implicants, and then less likely to happen.

5 Experiments

We have performed three series of experiments :

1. on controlled synthetic data
2. on OAEI directory benchmark
3. on Web Directories, with a comparison with the method SBI [7]

This section is structured according to these three series.

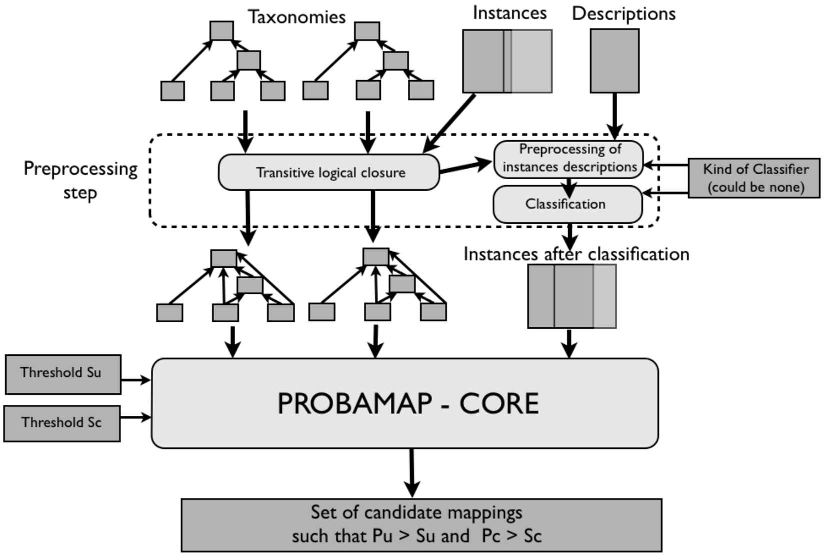

Taxonomies
Instances
Descriptions
Preprocessing step
Transitive logical closure
Preprocessing of instances descriptions
Classification
Kind of Classifier (could be none)
Instances after classification
Threshold Su
Threshold Sc
PROBAMAP - CORE
Set of candidate mappings such that Pu > Su and Pc > Sc

Fig. 4. Structure of ProbaMap

5.1 Experiments on synthetic data

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 lines 14 and 11 $Implicant(m)$ (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 [23]. 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.

In order to measure the robustness we have tested the robustness of Algorithm 1 to noisy data.

Finally, we have conducted experiments on real-world data to check the scalability of the Algorithm 1 on large taxonomies and to make qualitative measurements.

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 on synthetic and real-world data.

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 [24]. 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 [25], 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 [22] 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 A_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 5.1 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. Data generation in our context represents quite challenging issue when compared to other synthetic data generation, such as functional Armstrong relation for functional dependencies[5] [26], or transactional databases for frequent itemsets [27].

[5] An Armstrong relation for a set of functional dependencies is a relation that satisfies each dependency implied by the set and does not satisfy any dependency that is not implied by it.

Roughly speaking, we borrow the same principles than those developed for Armstrong relations [28]. Each generated mapping should be satisfied by the generated data, and each mapping that is not generated should be contradicted by the generated data.

With regard to the Armstrong relations, our generation tackles additional issues. Firstly, the structure may be generated whereas Armstrong relations suppose the structure (schemas) given. Secondly, for relational databases, it is enough to generates two tuples that contradict one dependency for ensuring that the dependency is not satisfied. In our case where mapping probabilities are estimated from statistics on class extensions, the amount of instances that contradict a mapping has a strong impact on its validity, then we can not only generate one instance for each mapping to be contradicted.

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

For an overview of the complete setting, the connection between ProbaMap and the generator is pictured in Figure 5. Distinguishing between the parameters of ProbaMap (two thresholds for P_c and P_u and the kind of classifier) and thoses of the generator is a key point.

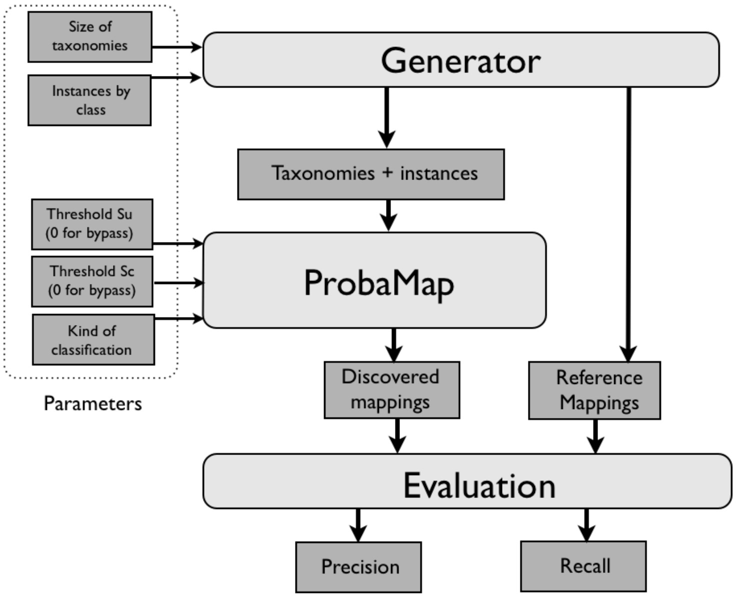


Fig. 5. Structure of ProbaMap

The first goal is to analyze the impact on the precision of the thresholds S_c, S_u, 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 three validity criteria the one offering the best performances : using P_u alone, using P_c alone, or using both of them.

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.

The last goal with synthetic data is to analyse the robustness of the approach to noisy data.

For all the experiments on synthetic data 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.

Experimental results 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 synthetic 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 6.

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 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.

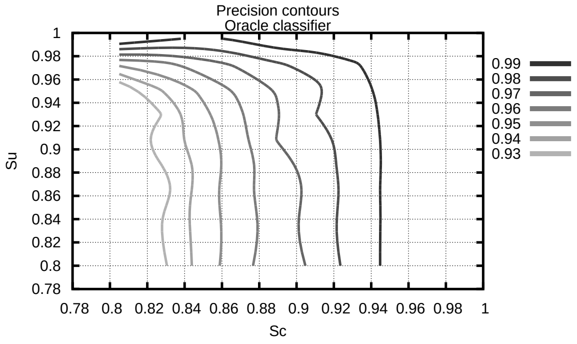


Fig. 6. Precision w.r.t. S_c and S_u thresholds

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.

Impact of probabilities on pruning ratio

We now study the impact of three probabilistic criteria for testing the validity of a mapping on the pruning ratio performed by Algorithm 1 :

1. using only P_u by setting $S_c = 0$ (bypass)
2. using only P_c by setting $S_u = 0$ (bypass)
3. using both P_u and P_c

Figure 7 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 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

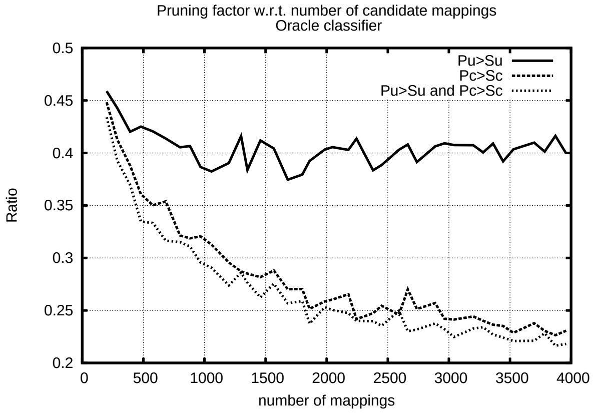


Fig. 7. Pruning factor for validity computation with $\widehat{P_c}$, $\widehat{P_u}$ and $\widehat{P_c}$ *and* $\widehat{P_u}$

$\widehat{P_c}$ 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. For example, a search space of 3500 mappings is pruned up to 22% by combining P_c and P_u, that implies that there are only 770 examinated mappings for which P_u and P_c are estimated.

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 [20], C4.5 [29] and SVM [30]. We use the Weka implementation of these classifiers and have interfaced it with our code.

The comparisons of running times are shown in Figure 8 and in log scale in Figure 9.

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 10 and Figure 11.

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 12, 13, and 14

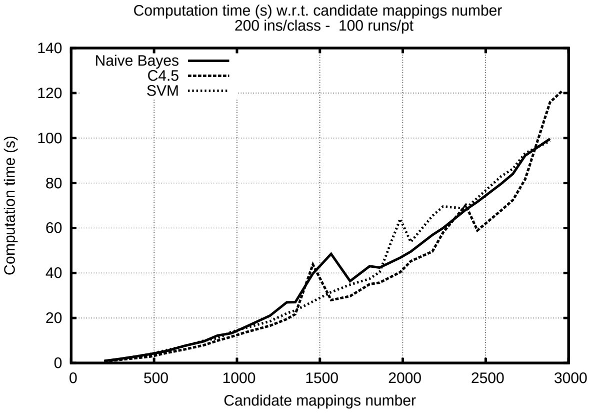


Fig. 8. Computation time (s) for different classifiers

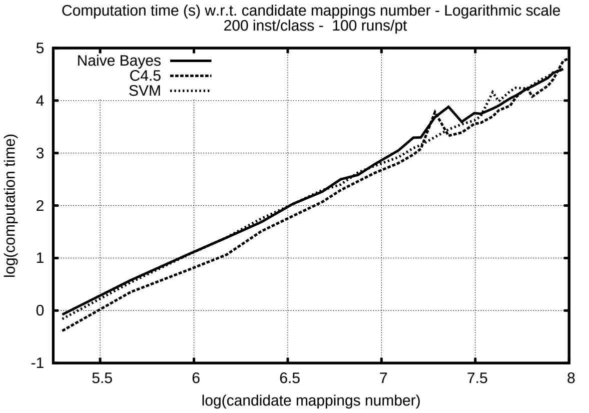


Fig. 9. Computation time in log scale for different classifiers

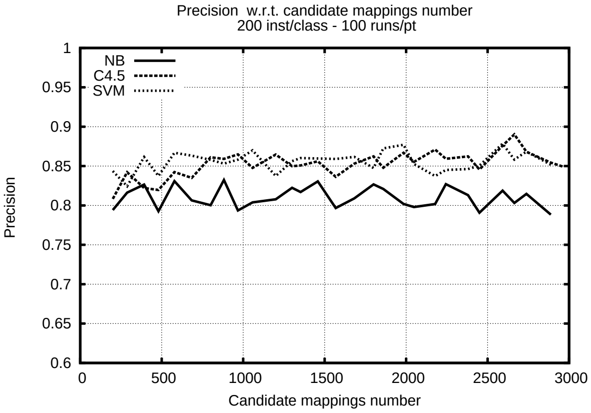


Fig. 10. Precision for different classifiers

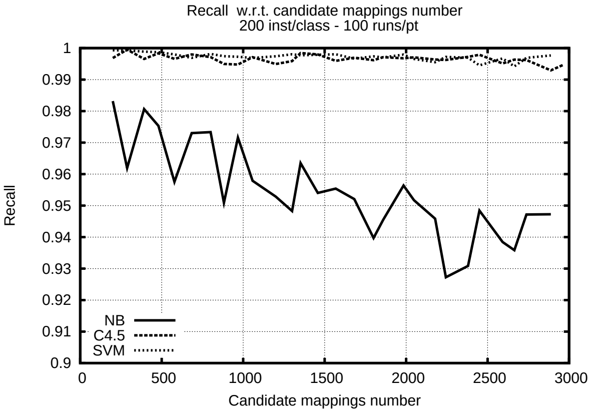


Fig. 11. Recall for different classifiers

In this experiment, the number of classes and of mapping is constant, hence the number of classifications to perform is linear in the number of instances. The C4.5 algorithm takes linear time in the number of instances. In fact, all instances are classified into each class at the beginning of ProbaMap, this is also perfectly coherent. This linearity 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 Bayesian approach of statistics.

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

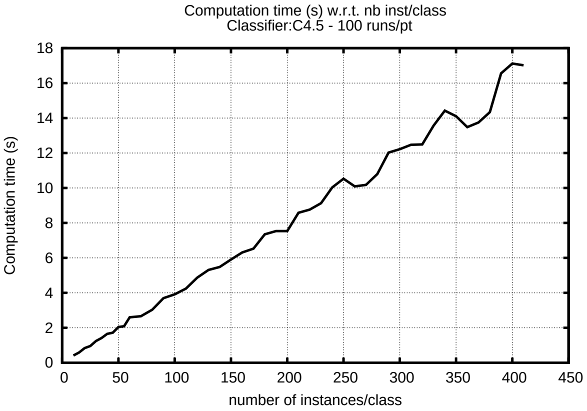


Fig. 12. C4.5 - Computation time (s) - impact of number of inst/class

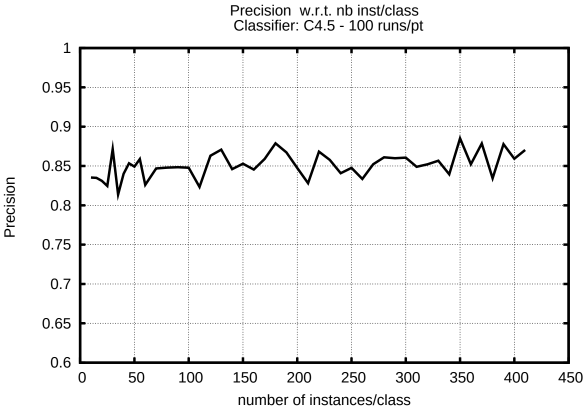


Fig. 13. C4.5 - Precision - impact of number of inst/class

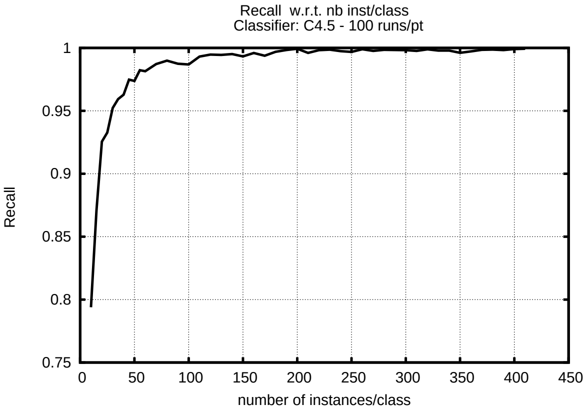


Fig. 14. C4.5 - Recall - impact of number of inst/class

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 15.

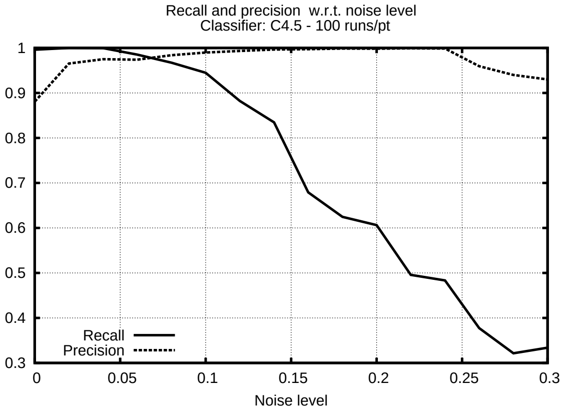


Fig. 15. C4.5 - Precision and recall - impact of noise level

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.

5.2 Real-world OAEI data

We have made experiments on the directory set of OAEI [12, 13] contest. This set is constituted by two large taxonomies of respectively 2857 and 6628 classes. For the contest, due to scalability issues, the taxonomies are split into the set of their branches, a small subset of which is given to the competitors for mapping alignment. In contrast, our algorithm is able to handle the two whole taxonomies, thus taking advantage of the complete structure of the taxonomies. It is important to note that without pruning, this would lead to a search space of 30 million mappings.

For compensating the absence of available instances for these taxonomies, we use a method inspired by [14] to automatically populate the classes with synsets[6]. We follow a similar method to Ctx-Match [14]. Intuitively, the principle of this population is based on associating each class C with a set of WordNet synsets (atomistic semantic unit in this thesaurus) that reflect the right meaning of the label of C in the context where it appears, i.e. the labels of the ancestor classes of C in its taxonomy. This help to disambiguate the meaning of a word: for instance, the label “Arizona” can correspond to a state of the U.S.A. or to a snake. If the Arizona class is a child class of “Animals”, the label “Animals” can be used to guess that “Arizona” means the snake species.

On the two whole taxonomies, the population phase produces about 30000 instances and takes 5 hours while the mapping discovery algorithm itself only takes 11 minutes. These are very reasonable computational times for handling 30 million possible mappings.

For evaluating the precision of the set of mappings discovered by our algorithm, we could only compute a lower bound based on the partial reference provided by OAEI. The results are promising, as for the thresholds S_u and S_c respectively set to 0.9 and 0.8 we obtained a lower bound of precision of 67%[7].

[6] In WordNet, a synset is a semantic unit, 2 synonyms correspond to the same synset.

[7] Fore more details, see `http://disi.unitn.it/~pane/OAEI/2009/directory/result/`

5.3 Comparative analysis on collected Web Directories

In this section, we test ProbaMap on part of Internet directories from Yahoo! and Google (actually based on Dmoz) that are rooted with similar or very close label. These sub-directories are considered as taxonomies, and URLs referenced inside each class of the taxonomy as instances.

The main difference with the sequence of experiments in the previous section is that the dataset contains original instances that are collected with their taxonomies, avoiding to process an artificial population.

We compare our approach to the SBI algorithm of Ichise et al. [7, 31], which is dedicated to the discovery of mappings between Internet directories, and the integration of such directories.

Internet directories are trees of categories, which can be seen as taxonomies, categories being the classes. Each category contains a set of links (i.e. URLs to web sites), which can be seen as the instances of the class. Each link comes with a small text summary, whose words can be seen as instance attributes for classification.

Our datasets are corresponding locations in the Yahoo! and Google directories, that have also been used in the experiments of [7, 31]:

- Yahoo! : `Recreation / Automotive` & Google : `Recreation / Autos`
- Yahoo! : `Recreation / Outdoors` & Google : `Recreation / Outdoors`
- Yahoo! : `Computers_and_Internet/Software` & Google : `Computers/Software`
- Yahoo! : `Arts / Visual_Arts / Photography` & Google : `Arts / Photography`

The data from the directories was collected in June 2010, so is different from the data of [31] and [7] which was collected in Fall 2001.
Table 1 shows for each dataset the number of classes and instances in each class, and the number of instances shared between the Yahoo! and the Google directories. Two instances are considered shared if they correspond to the same URL in both directories. For a fair comparison, we have implemented both ProbaMap and the SBI algorithm in Java.

	Yahoo!		**Google**		shared
	classes	instances	classes	instances	instances
`Autos`	947	4406	967	6425	837
`Outdoors`	2428	5511	1441	13863	623
`Software`	323	2390	2395	30140	572
`Photography`	168	1851	321	3852	286

Table 1. Statistics on data collected from subdirectories on Yahoo! and Google

Experimental protocol An overview of the setting used for the comparative experiment is pictured in Figure 16.

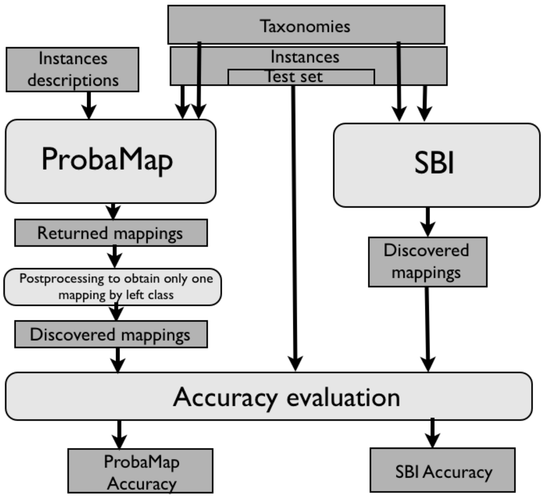


Fig. 16. Setting used for the comparative experiment between SBI and ProbaMap

SBI takes as input one source taxonomy, one target taxonomy, and the instances declared for each class of both taxonomies. For each class C_s of the source taxonomy, SBI returns a rule $C_s \rightarrow C_t^{predicted}$ associating C_s to a target class $C_t^{predicted}$ in the target taxonomy.

In order to fit the evaluation framework of SBI, we added a postprocessing to ProbaMap to obtain a similar form of results, i.e. a set of unique rules for each class of the source taxonomy.

The complete process is the following:

1. Application of ProbaMap on $\mathcal{T}_1$ and $\mathcal{T}_2$
2. For each class C_1 of $\mathcal{T}_1$,
   among all C_2 for which the two mappings $C_1 \sqsubseteq C_2$ and $C_2 \sqsubseteq C_1$ have been discovered,
   select the class C_2 for which $min(\widehat{P_c}(C_1 \sqsubseteq C_2), \widehat{P_c}(C_2 \sqsubseteq C_1))$ has the highest value.
3. For each class C_1 of $\mathcal{T}_1$, if there is no rule for C_1, associate to C_1 the rule of its closest ancestor in $\mathcal{T}_1$

In this way we obtain an unique rule $C_1 \rightarrow C_2$ for each class of $\mathcal{T}_1$, like the SBI system.

As there are a good proportion of shared instances for each directory that makes the classification step not mandatory, we can and we will compare SBI against both ProbaMap versions with and without the classification step. For the version without classification, preliminary tests lead to set $S_c = 0.6$ and $S_u = 0.9$. SVM is set to be the default classifier by using the SMO implementation in weka. When using classification, because of the small values of the joint probabilities of classes, we set $S_u = S_c = 0$. In this case, the postprocessing step is modified to take into account the probabilities P_c and P_u of the reverse mappings.

Comparative qualitative evaluation The goal of our experiments is to compare the quality of Internet directories alignment for ProbaMap and SBI.

For the discovery of mappings, ProbaMap and SBI receive a "training" set of instances which is a subset of the shared and annotated instances. The test set used for evaluation of the discovered mappings is constituted by the remaining instances among the shared ones. In the case where ProbaMap is set to use classification, the training set is extended with all the non shared instances. The ratio of the number of shared instances used for training among all shared instances is a controlled variable on the experiments we have conducted.

The classification is performed using the SVM implementation SMO [30] in Weka [21], where the classification attributes for an instance are the words of its summary. The experiments on controlled data have shown that SVM and C4.5 are comparable in quality but that C4.5 is faster than SVM. Nevertheless, we have conducted this experiment with SVM which is expected to perform better on sparse data.

The evaluation is done by using the test set of instances. Each instance of this set belongs to a class C_s of the source taxonomy. Hence, we can compare:

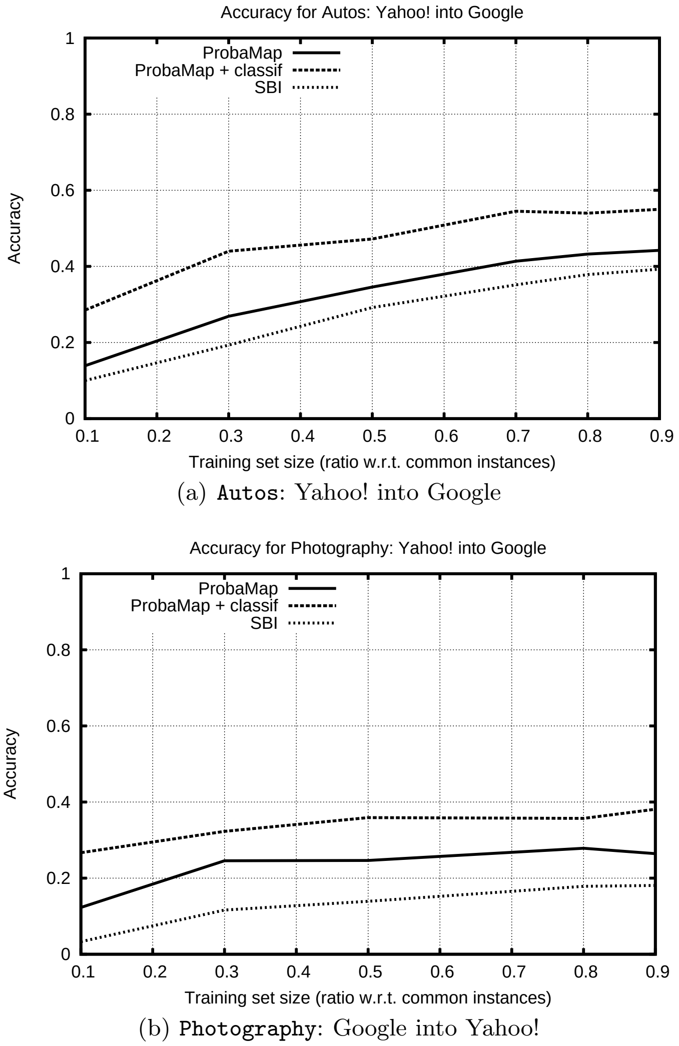
- the class $C_t^{predicted}$ of the instance, predicted by the output rule $C_s \rightarrow C_t^{predicted}$
- the class C_t in which the instance is declared in the target taxonomy (each instance of the test set is common to the two taxonomies)

The *accuracy* measures the ratio of the instances in the test set for which $C_t^{predicted} = C_t$. Accuracy is a standard micro-averaged evaluation measure which is based on instances, whereas recision and recall are macro-averaged measures based on mappings themselves. As there is no reference of mappings provided for the considered Yahoo! and Google subdirectories, but a sufficient proportion of instances are shared by them, we use the accuracy criterion to evaluate ProbaMap in this experiment. This enables also to fit the evaluation framework of SBI (described in [31]).

Ratio of shared instances provided for training	**SBI**	**ProbaMap**	**ProbaMap + classif**
0.5	0.23	0.28	0.36
0.9	0.29	0.33	0.40

Table 2. Averaged accuracy for SBI and ProbaMap

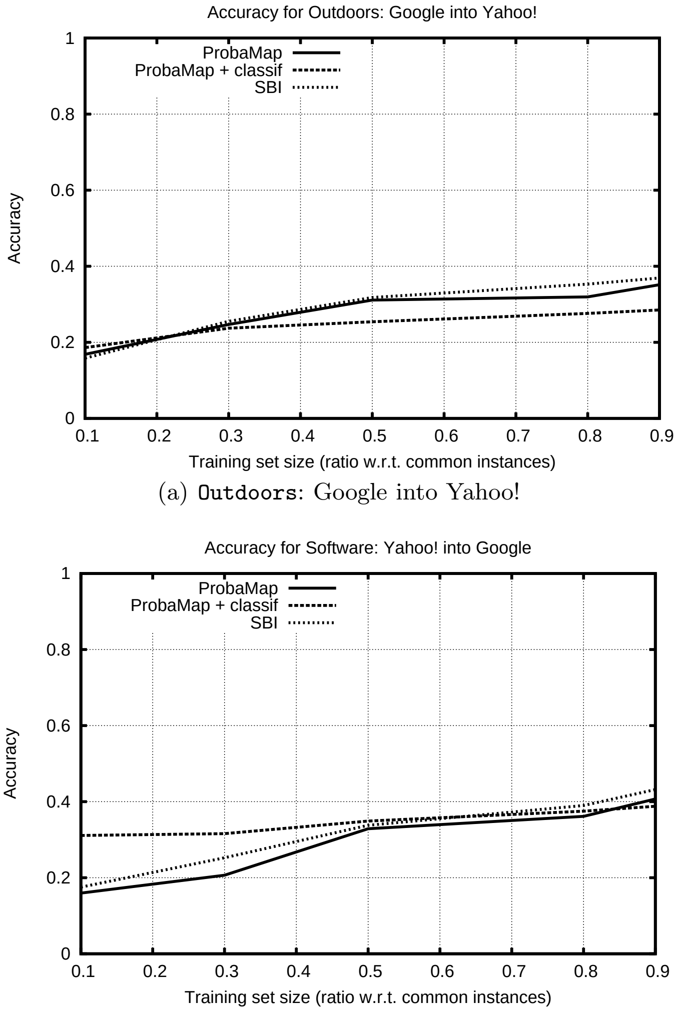
Results The averaged results in Table 2 show that ProbaMap outperforms SBI in average, and that ProbaMap with classification outperforms ProbaMap



(a) `Autos`: Yahoo! into Google

(b) `Photography`: Google into Yahoo!

Fig. 17. Comparative accuracy results (1)



(a) `Outdoors`: Google into Yahoo!

(b) `Software`: Google into Yahoo!

Fig. 18. Comparative accuracy results (2)

without classification. The average is computed on the four directories of the Table 1 and for each direction of alignment.

In particular, ProbaMap with classification significantly outperforms ProbaMap without Classification (about 10% better) and SBI (about 20% better) for the datasets `Autos` and `Photography`, whatever the size of the training set.

For the directories `Software` and `Outdoors`, ProbaMap without classification and SBI both provide lower results. SBI performs a little better (no more than 5%). In these two cases, there are initially few instances by class, and then the classification step allows to improve the results : ProbaMap with classification outperforms SBI on the `Software` directory up to 60% of instances for the training set, whereas ProbaMap without classification does not. For the `Outdoors` results pictured in Figure 18(a), ProbaMap with classification is better for a small training set ($\leq$ 20% of the shared instances). The reason is that, for this particular dataset, a very small ratio of instances is classified and then the extensions of classes do not change much after the classification step.

We have also checked that all these results both for ProbaMap and ProbaMap with classiffcation are rougly better in average than the results of a modified version of SBI using Naive Bayes classification so-called SBI-NB [31]. More precisely, both versions of Probamap perform better for 6 of the 8 tested directories. Note however that we are provided results for SBI-NB with an old dataset collected in 2004, so the comparison is not totally fair.

Finally, SBI and ProbaMap (without classification) both take a few seconds on each of the directories of Figure 1). The version of ProbaMap with classification requires additionnal time (several minutes) for classifying each instance into each class, but this significantly improves the quality of the results.

These experiments show that the mapping discovery method that we propose gives good results on real-world datasets, and can take advantage of classification techniques to compensate small training set sizes. This is an important quality for real world taxonomies built by different people, that are unlikely to have many instances in common.

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 [32]. 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 [33], H-MATCH [11]) . Another widely used principle is structure comparison between labeled graphs representing ontologies (e.g., OLA [9]), based on similarity flooding [34]. The underlying idea is that two elements of two distinct ontologies are similar when their adjacent elements are similar. This is applied by spreading similarities in the spirit of how IP packets flood the network in broadcast communication.

A category of methods (e.g., FCA-merge [35], [6]) are called "extensional" because they are instance-based: they rely on instances of classes to compute matches between them. It is shown in [36] that the reliability of instance-based methods depends on the applications and on the kind of correspondences that are considered. The instance-based method SBI [7] with which we have compared our method is based on the computation of the Fleiss Kappa coefficients that are symmetric and then they cannot be interpreted as probabilities on inclusion mappings.

Most of the instance-based methods make use of classifiers using a corpus of labelled instances (e.g., LSD [8], SemInt [37], GLUE [5],[38]). It should be pointed out that GLUE discovers symmetric correspondences that have no formal semantics. The associated confidence value for each discovered correspondence are not real probabilities as they are based on similarities like the Jaccard similarity. In contrast, we consider mappings that denote inclusions between classes of different taxonomies, and we define a formal probabilistic semantics for these inclusion mappings.

sPLmap [38] is an approach for schema matching, and oPLmap [39] is a variant for ontology matching, that rely on classifiers and deal with probabilities. sPLMap finds the best set of mappings M between two schemas S,T that maximizes the probability that the tuples in the rewritten schema S using M (denoted S_M) are plausible in T and vice versa. At the end, each returned mapping is associated with a probability based on the conditional probability formula. In contrast with our work, sPLmap computes probabilities by combining scores provided by several weighted classifiers (working on text or data values of tuples).

Other approaches have been investigated with machine learning techniques using a corpus of schema matches (e.g., [40],[41]). The work introduced in [41] uses classifiers directly on *correspondences*, by representing each correspondence in a vector space constructed from instances features. A training set of true correspondences should be provided. Then, for a tested correspondence between two classes A and B, the similarities between (i) the instances of A, (ii) the instances of B, and (iii) the instances of all examples correspondences allow to give to the tested correspondence a position in the correspondence space. Hence the classifier can be used to determine if the tested correspondence is relevant or not, according to its position in the correspondence space and the learned examples.

In fact, most of the existing matchers combine these elementary approaches in different ways (e.g., COMA++ [42] and COMA [43], Cupid [10], H-MATCH [11], Lily [44], S-Match [45], Clio [46]).

In [47], an extension of existing matching methods is proposed by considering the k ranked best alignments. The top-k alignment ranking is combined with the schema to match in order to generate the final alignment that globally maximizes its score.

It is standard practice for ontology and schema matchers to associate numbers with the candidate mappings they propose. The uncertainty is intrinsic to correspondences discovery because two classes or properties (for example) independently created are unlikely to exactly match. As stated in [15], there is still a need to better understand the foundations of modeling uncertainty that is primary important for improving the detection of correspondences causing inconsistencies, e.g., via probabilistic reasoning, or to identify where the user feedback is maximally useful, and for improving the quality of the interpretation of correspondences.

However, those uncertainty coefficients 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 [5], we use them for computing true probabilities and not similarity coefficients.

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., Ctx-Match [14], Clio [46]). The work in [48] introduces a framework for modeling and evaluating automatic semantic reconciliation. It provides a formal model for semantic reconciliation and theoretically analyses the factors that impact the effectiveness of matching algorithms. The proposed formal model borrows from fuzzy set theory for handling uncertainty by combinations of similarity measures and good properties. This work is not connected probability but is complementary to the approach in this thesis.

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 [49]. More generally, our approach is complementary of the recent work that has been flourishing on probabilistic databases [50, 51]. In particular, it fits into the general framework set in [16] for handling uncertainty in data integration, for which it provides an effective way for computing mapping probabilities. ProbaMap can be generalized to perform without instances at all, by using techniques that lead to correctly estimate $P(A_1)$ and $P(A_1 \cap B_2)$ (for a mapping $A_1 \sqsubseteq B_2$). Such a generalization should take care that the distributions of A and of $A \cap B$ are considered and estimated in a probabilistic framework. For example, a linguistic resource can be used for estimating thoses probabilities by taking the labels of A and B and returning *probabilities* for A and $A \cap B$, not only coefficients. This requires that the techniques to obtain such probabilities from the linguistic resource should respect some properties that fit with the probability theory.

The experiments that we have conducted on both real-world and controlled synthetic data have shown the feasibility and the scalability of our approach. In contrast with our approach that prunes the search space by an appropriate ordering for generating and testing the mappings, a lot of existing approaches compute a similarity matrix for all pairs of classes. The ordering used in SBI is not logically consistent. Several complex matching system use the logical consistency as a filter or a way to check their results, but not for pruning the search space. Up to now, the OAEI context does not focus on scalability, except for isolate cases like the `anatomy 2006` [8] dataset which contains large ontologies in the anatomy domain. Methods like PRIOR [52] and H-MATCH [53] has shown good performances on this dataset. The PRIOR system makes use of information retrieval techniques, by indexing a profile for each class (or any ontology entity), and by using a query answering algorithm to rank and find the best matches for a particular class by making a query about its own profile. Note that the anatomy dataset does not contain any instance.

Another possible approach investigated in [54] for handling large taxonomies is based on partitioning. Our approach is scalable while keeping the whole logical structure, which can be potentially lost by partitioning taxonomies.

Perspectives

We envision two main perspectives. First, we will study a setting for probabilistic query answering, in the spirit of probabilistic databases [16]. In such a setting, probabilities of mappings will be used by the query answering algorithm to give some probability values for each answer. In particular, it should be interesting to focus on reasoning-based query algorithm like in the Somewhere [55] setting. There are existing works on introducing probabilities in logic and inference process. Probability Logic [17] fits our proposed model in the way that the classical implication and the conditional are both present in the language. Probabilistic description logics ([56], [49]) are also based on conditional formulas too. Based on inference rules and their properties about probabilities, such probabilistic and logical framework can be extended to do probabilistic reasoning involving probabilistic mappings.

Our second pespective tackles the central issue in ontology matching (see [15]), that is to provide a formal semantics to coefficients returned as output of existing alignment methods. The idea is to design and implement a post-processing step in order to transform the returned coefficients into coefficients that can be interpreted as probabilities, i.e. that respect the property of monotony (Theorem 1). For this purpose, we plan to use the similarity flooding principle [34], in the spirit of N2R [57] and OLA [9]. Coefficients for mappings are initialized by those returned by the existing method to postprocess. The properties of monotony are then translated into strong influences between coefficients of mappings connected by an entailment relation. These influences between mappings coefficients are finally modeled by (non-linear) equations involving the maximum function. Like in N2R, the main issue would be to find an iterative algorithm ensured to converge towards a fixpoint that is the solution of the equation system.

References

1. Dean, M., Schreiber, G.: OWL web ontology language reference. W3C recommendation, W3C (February 2004)

[8] http://oaei.ontologymatching.org/2006/anatomy/

2. Hayes, P., ed.: RDF Semantics. W3C Recommendation. World Wide Web Consortium (February 2004)
3. Rahm, E., Bernstein, P.A.: A survey of approaches to automatic schema matching. VLDB J. **10**(4) (2001) 334–350
4. Shvaiko, P., Euzenat, J.: A survey of schema-based matching approaches. J. Data Semantics IV (2005) 146–171
5. Doan, A., Madhavan, J., Domingos, P., Halevy, A.Y.: Learning to map between ontologies on the semantic web. In: WWW. (2002) 662–673
6. David, J., Guillet, F., Gras, R., Briand, H.: An interactive, asymmetric and extensional method for matching conceptual hierarchies. In: EMOI-INTEROP Workshop, Luxembourg. (2006)
7. Ichise, R., Takeda, H., Honiden, S.: Integrating multiple internet directories by instance-based learning. In: international joint conference on artificial intelligence. Volume 18. (2003) 22–30
8. Doan, A., Domingos, P., Levy, A.Y.: Learning mappings between data schemas. In: Proceedings of the AAAI-2000 Workshop on Learning Statistical Models from Relational Data. (2000)
9. Euzenat, J., Valtchev, P.: Similarity-based ontology alignment in owl-lite. In: ECAI. (2004) 333–337
10. Madhavan, J., Bernstein, P.A., Rahm, E.: Generic schema matching with cupid. In: The VLDB Journal. (2001) 49–58
11. Castano, S., Ferrara, A., Montanelli, S.: H-match: an algorithm for dynamically matching ontologies in peer-based systems. In: SWDB. (2003) 231–250
12. Euzenat, J.: Ontology alignment evaluation initiative (July 2008) `http://oaei.ontologymatching.org/`.
13. Euzenat, J., Ferrara, A., Hollink, L., Isaac, A., Joslyn, C., Malais, V., Meilicke, C., Nikolov, A., Pane, J., Sabou, M., et al.: Results of the ontology alignment evaluation initiative 2009. In: Fourth International Workshop on Ontology Matching, Washington, DC. (2009)
14. Serafini, L., Bouquet, P., Magnini, B., Zanobini, S.: An algorithm for matching contextualized schemas via SAT. Proceedings of CONTEXT'03 (2003)
15. Shvaiko, P., Euzenat, J.: Ten challenges for ontology matching. In Meersman, R., Tari, Z., eds.: On the Move to Meaningful Internet Systems: OTM 2008. Springer Berlin / Heidelberg (2008) 1164–1182
16. Dong, X.L., Halevy, A.Y., Yu, C.: Data integration with uncertainty. In: VLDB. (2007) 687–698
17. Adams, E.: A Primer of Probability logic, CSLI. Stanford University, Stanford, California (1998)
18. Tournaire, R., Rousset, M.C.: Découverte automatique de correspondances entre taxonomies - internal report (in french) (2008) http://membres-liglab.imag.fr/tournaire/irap08.pdf.
19. Degroot, M.H.: Optimal Statistical Decisions (Wiley Classics Library). Wiley-Interscience (April 2004)
20. Mitchell, T.: Machine Learning. McGraw-Hill Education (ISE Editions) (1997)
21. Witten, I.H., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques. 2 edn. Morgan Kaufmann (2005)
22. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms, Second Edition. The MIT Press (September 2001)
23. Van Rijsbergen, C.J.: Information retrieval / C. J. van Rijsbergen. Butterworths, London ; Boston : (1975)

24. Duchon, P., Flajolet, P., Louchard, G., Schaeffer, G.: Boltzmann samplers for the random generation of combinatorial structures. Comb. Probab. Comput. **13**(4-5) (2004) 577–625
25. Euzenat, J.: Semantic precision and recall for ontology alignment evaluation. In: IJCAI. (2007) 348–353
26. Beeri, C., Dowd, M., Fagin, R., Statman, R.: On the structure of armstrong relations for functional dependencies. Journal of the ACM (JACM) **31**(1) (1984) 30–46
27. Ramesh, G., Maniatty, W., Zaki, M.J.: Feasible itemset distributions in data mining: theory and application. In: PODS. (2003) 284–295
28. Fagin, R.: Horn clauses and database dependencies. J. ACM **29**(4) (1982) 952–985
29. Quinlan, R.J.: C4.5: Programs for Machine Learning (Morgan Kaufmann Series in Machine Learning). Morgan Kaufmann (January 1993)
30. Flake, G.W., Lawrence, S.: Efficient svm regression training with smo. Mach. Learn. **46**(1-3) (2002) 271–290
31. Ichise, R., Hamasaki, M., Takeda, H.: Discovering relationships among catalogs. In Einoshin Suzuki and Setsuo Arikawa, Editors, Discovery Science **3245** (2004) 371–379
32. Euzenat, J., Shvaiko, P.: Ontology matching. Springer-Verlag, Heidelberg (DE) (2007)
33. Hamdi, F., Zargayouna, H., Safar, B., Reynaud, C.: TaxoMap in the OAEI 2008 alignment contest . In: Ontology Alignment Evaluation Initiative (OAEI) 2008 Campaign - Int. Workshop on Ontology Matching. (2008)
34. Melnik, S., Garcia-Molina, H., Rahm, E., et al.: Similarity flooding: A versatile graph matching algorithm and its application to schema matching. In: Proceedings of the International Conference on Data Engineering. (2002) 117–128
35. Stumme, G., Maedche, A.: FCA-MERGE: Bottom-Up Merging of Ontologies. In: Proc. of the 17^{th} International Joint Conference on Artificial Intelligence. (2001) 225–234
36. Isaac, A., van der Meij, L., Schlobach, S., Wang, S.: An empirical study of instance-based ontology matching. In: ISWC/ASWC. (2007) 253–266
37. Li, W.S., Clifton, C.: Semint: a tool for identifying attribute correspondences in heterogeneous databases using neural networks. Data Knowl. Eng. **33**(1) (2000) 49–84
38. Nottelmann, H., Straccia, U.: Information retrieval and machine learning for probabilistic schema matching. Information Processing and Management **43**(3) (2007) 552–576
39. Nottelmann, H., Straccia, U.: A probabilistic, logic-based framework for automated web director alignment. In Ma, Z., ed.: Soft Computing in Ontologies and the Semantic Web. Volume 204 of Studies in Fuzziness and Soft Computing. Springer Verlag (2006) 47–77
40. Madhavan, J., Bernstein, P.A., Doan, A., Halevy, A.: Corpus-based schema matching. International Conference on Data Engineering **0** (2005) 57–68
41. Wang, S., Englebienne, G., Schlobach, S.: Learning concept mappings from instance similarity. In: Proceedings of the 7th International Conference on The Semantic Web, Karlsruhe, Germany, Springer-Verlag (2008) 339–355
42. Aumueller, D., Do, H.H., Massmann, S., Rahm, E.: Schema and ontology matching with coma++. In: SIGMOD '05: Proceedings of the 2005 ACM SIGMOD international conference on Management of data, New York, NY, USA, ACM (2005) 906–908

43. Do, H.H., Rahm, E.: Coma - a system for flexible combination of schema matching approaches. In: VLDB. (2002)
44. Wang, P., Xu, B.: Lily: Ontology alignment results for OAEI 2009. Shvaiko et al.[SEG+ 09] (2009)
45. Giunchiglia, F., Shvaiko, P., Yatskevich, M.: S-match: an algorithm and an implementation of semantic matching. In: Proceedings of ESWS. (2004) 61–75
46. Chiticariu, L., Hernández, M.A., Kolaitis, P.G., Popa, L.: Semi-automatic schema integration in clio. In: VLDB. (2007) 1326–1329
47. Gal, A.: Managing uncertainty in schema matching with top-k schema mappings. Journal on Data Semantics **6** (2006) 2006
48. Gal, A., Anaby-Tavor, A., Trombetta, A., Montesi, D.: A framework for modeling and evaluating automatic semantic reconciliation. The VLDB Journal **14**(1) (2005) 50–67
49. Castano, S., Ferrara, A., Lorusso, D., Nth, T.H., Möller, R.: Mapping validation by probabilistic reasoning. In: Proc. 5th European Semantic Web Conference. (2008)
50. Benjelloun, O., Sarma, A.D., Halevy, A.Y., Widom, J.: Uldbs: Databases with uncertainty and lineage. In: VLDB. (2006) 953–964
51. Dalvi, N.N., Suciu, D.: Answering queries from statistics and probabilistic views. In: VLDB. (2005) 805–816
52. Mao, M., Peng, Y.: PRIOR system: Results for OAEI 2006. Proceedings of the Ontology Alignment Evaluation Initiative (2006) 165–172
53. Castano, S., Ferrara, A., Messa, G.: Results of the HMatch ontology matchmaker in OAEI 2006. In: Proceedings of the ISWC 2006 Workshop on Ontology Matching, Athens, GA, USA. (2006)
54. Hamdi, F., Safar, B., Reynaud, C., Zargayouna, H.: Alignment-based Partitioning of Large-scale Ontologies. In Henri Briand, Fabrice Guillet, Gilbert Ritschard, Djamel Zighed, eds.: Advances in Knowledge Discovery And Management. Springer (2009)
55. Adjiman, P., Chatalic, P., Goasdoué, F., Rousset, M.C., Simon, L.: Distributed reasoning in a peer-to-peer setting: Application to the semantic web. J. Artif. Intell. Res. (JAIR) **25** (2006) 269–314
56. Koller, D., Levy, A., Pfeffer, A.: P-CLASSIC: a tractable probablistic description logic. In: Proceedings of the National Conference on Artificial Intelligence. (1997) 390–397
57. Saïs, F., Pernelle, N., Rousset, M.C.: Combining a logical and a numerical method for data reconciliation. Journal on Data Semantics XII (2009) 66–94
58. Tournaire, R., Petit, J.M., Rousset, M.C., Termier, A.: Discovery of probabilistic mappings between taxonomies: Principles and experiments (technical report) (2009) http://membres-liglab.imag.fr/tournaire/longpaper.pdf.
59. Fellbaum, C.: WordNet: An Electronic Lexical Database (Language, Speech, and Communication). The MIT Press (May 1998)
60. Nilsson, M., O'Neill, D.: Id3 official site `http://www.id3.org/`.
61. Lin, F., Sandkuhl, K.: A survey of exploiting wordnet in ontology matching. Artificial Intelligence in Theory and Practice II (2008) 341–350
62. Resnik, P.: Semantic similarity in a taxonomy: An information-based measure and its application to problems of ambiguity in natural language. Journal of artificial intelligence research **11**(95) (1999) 130