A Probabilistic, Logic-based Framework for Automated Web Directory Alignment

Henrik Nottelmann[1] and Umberto Straccia[2]

[1] Institute of Informatics and Interactive Systems, University of Duisburg-Essen, Duisburg, Germany `nottelmann@uni-duisburg.de`
[2] ISTI-CNR, Pisa, Italy `straccia@isti.cnr.it`

Summary. We introduces oPLMap, a formal framework for automatically learning mapping rules between heterogeneous Web directories, a crucial step towards integrating ontologies and their instances in the Semantic Web. This approach is based on Horn predicate logics and probability theory, which allows for dealing with uncertain mappings (for cases where there is no exact correspondence between classes), and can be extended towards complex ontology models. Different components are combined for finding suitable mapping candidates (together with their weights), and the set of rules with maximum matching probability is selected. Our system oPLMap with different variants has been evaluated on a large test set.

1 Introduction

While the World Wide Web has been merely a collection of linked text and multimedia documents, it is currently evolving into documents with semantics, the *Semantic Web*. In this context, ontologies, which have been studied intensively for a long time, become more and more popular. Ontologies are formal definitions of concepts and their relationships. Typically, concepts are defined by classes, which are organised hierarchically by specialization (inheritance) relationships. A simple example is a Web directory, which consists of a simple class hierarchy. For example, the concept "Modern History" in Fig. 1 is a specialization (a sub-class) of the concept "History" [3].

With the emergence of ontologies and their instances in the Semantic Web, their heterogeneity constitutes a new, crucial problem. The Semantic Web is explicitly built upon the assumption that there is no commonly used ontology for all documents; instead, the Semantic Web will be populated by many different ontologies even for the same area. Thus, mapping/alignment between

[3] RDF Schema, and the OWL family of languages (OWL Full, OWL DL and OWL Lite) [21] are becoming major ontology definition languages in the Semantic Web. The latter ones are related to *Description Logics* [1], which allow for defining also properties of instances in addition to concepts

different ontologies becomes an important task. For instance, an excerpt of two "course" ontologies is given in Fig. 1. It also reports the mappings between the classes of the two ontologies. Of course, finding out these mappings automatically is desirable.

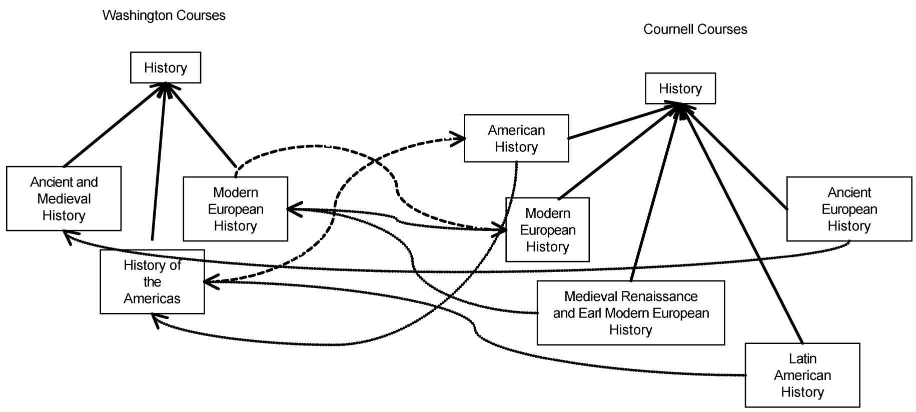

Fig. 1. The excerpt of two ontologies and class matchings

This paper proposes a new approach, called *oPLMap*, for automatically learn the mappings among tree like ontologies, e.g. web directories. Web directory alignment is the task of learning mappings between heterogeneous Web directory classes. Our approach is based on a logical framework, which is combined with probability theory (probabilistic Datalog), and aims at finding the optimum mapping (the mapping with the highest matching probability). It borrows from other approaches like GLUE [11] the idea of combining several specialized components for finding the best mapping. Using a probabilistic, logic-based framework bears some nice features: First, in many cases mappings are not absolutely correct, but hold only with a certain probability. Defining mappings by means of probabilistic rules is a natural solution to this problem. Second, classes have often attributes (called properties). These properties can easily be modelled by additional Datalog predicates. In this paper, however, we restrict to a fairly simply model where only textual content is considered.

The paper is structured as follows: The next section introduces a formal framework for learning the mappings, based on a combination of predicate logics with probability theory. Section 3 presents a theoretically founded approach for learning these mappings, where the predictions of different classifiers are combined. Our approach is evaluated on a large test bed in section 4. The last section summarizes this paper, describes how this work is related to other approaches and gives an outlook over future work.

2 Web directory alignment

This section introduces a formal, logics-based framework for Web directory alignment. It starts from the formal framework for information exchange in [14] and extents it to a framework cable to cope with the intrinsic uncertainty of the mapping process. The framework is based on probabilistic Datalog [17], for which tools are available. The mapping process is fully automatic.

2.1 Probabilistic Datalog

In the following, we briefly describe Probabilistic Datalog (pDatalog for short) [17]. pDatalog is an extension to Datalog, a variant of predicate logic based on function-free Horn clauses. Negation is allowed, but its use is limited to achieve a correct and complete model. However, for ease of presentation we will not deal with negation in this paper. In pDatalog every fact or rule has a probabilistic weight $0 < \alpha \leq 1$ attached, prefixed to the fact or rule:

$$\alpha \ A \leftarrow B_1, \ldots, B_n \ .$$

Here, A denotes an atom (in the rule head), and $B_1, \ldots, B_n$ ($n \geq 0$) are atoms (the sub goals of the rule body). A weight $\alpha = 1$ can be omitted. In that case the rule is called *deterministic*. For ease, a fact $\alpha \ A \leftarrow$ is represented as αA. Each fact and rule can only appear once in the program, to avoid inconsistencies. The intended meaning of a rule αr is that "the probability that an instantiation of rule r is true is α". For instance, assume that we have two web directories D_1 and D_2, the class "Aeronautics_and_Astronautics" belongs to D_1, while the class "Mechanical_and_Aerospace_Engineering" belongs to D_2. Then the following rule (mapping)

$$0.1062 \ \texttt{Mechanical\_and\_Aerospace\_Engineering(x)} \leftarrow \texttt{Aeronautics\_and\_Astronautics(x)} \ .$$

expresses the fact that a document about "Aeronautics_and_Astronautics" is also a document about "Mechanical_and_Aerospace_Engineering" with probability of 10.62% and, thus, establishes a bridge among the two web directories D_1 and D_2.

Formally, an *interpretation structure* is a tuple $\mathscr{I} = (\mathcal{W}, \mu)$, where $\mathcal{W}$ is a set of possible worlds and μ is a probability distribution over $\mathcal{W}$. The possible worlds are defined as follows. Given a pDatalog program P, with $H(P)$ we indicate the ground instantiation of P [4]. Then, the *deterministic part* of P is the set P_D of instantiated rules in $H(P)$ having weight $\alpha = 1$, while the *indeterministic part* of P is the set P_I of instantiated rules determined by

[4] The set of all rules that can be obtained by replacing in P the variables with constants appearing in P, i.e. the *Herbrand universe*.

$P_I = \{r : \alpha r \in H(P), \alpha < 1\}$. The set of deterministic programs of P, denoted $D(P)$ is defined as $D(P) = \{P_D \cup Y : Y \subseteq P_I\}$. Note that any $P' \in D(P)$ is a classical logic program. Finally, a possible world $w \in \mathcal{W}$ is the minimal model [26] of a deterministic program in $D(P)$ and is represented as the set of ground atoms that are true in the minimal model (also called *Herbrand model*). Now, an *interpretation* is a tuple $I = (\mathscr{I}, w)$ such that $w \in \mathcal{W}$. The truth of formulae w. r. t. an interpretation and a possible world is defined recursively as:

$$
\begin{aligned}
&(\mathscr{I}, w) \models A \text{ iff } A \in w \ , \\
&(\mathscr{I}, w) \models A \leftarrow B_1, \ldots, B_n \text{ iff } (\mathscr{I}, w) \models B_1, \ldots, B_n \Rightarrow (\mathscr{I}, w) \models A \ , \\
&(\mathscr{I}, w) \models \alpha r \text{ iff } \mu(\{w' \in \mathcal{W} : (\mathscr{I}, w') \models r\}) = \alpha \ .
\end{aligned}
$$

An interpretation $(\mathscr{I}, w)$ is a *model* of a pDatalog program P, denoted $(\mathscr{I}, w) \models P$, iff it entails every fact and rule in P:

$$(\mathscr{I}, w) \models P \text{ iff } (\mathscr{I}, w) \models \alpha r, \text{ for all } \alpha r \in H(P) \ .$$

In the remainder, given an n-ary atom A for predicate $\bar{A}$ and an interpretation $I = (\mathscr{I}, w)$, with A^I (an *instantiation* of A w. r. t. the interpretation A) we indicate the set of ground facts $\alpha\bar{A}(c_1, ..., c_n)$, where the ground atom $\bar{A}(c_1, ..., c_n)$ is contained in the world w, and $\mu(\{w' \in \mathcal{W} : (\mathscr{I}, w') \models \bar{A}(c_1, ..., c_n)\}) = \alpha$, i.e. $I \models \alpha\bar{A}(c_1, ..., c_n)$. Essentially, A^I is the set of all instantiations of A under I with relative probabilities, i.e. under I, $\bar{A}(c_1, ..., c_n)$ holds with probability α. Finally, given a ground fact αA, and a pDatalog program P, we say that P *entails* αA, denoted $P \models \alpha A$ iff in all models I of P, $I \models \alpha A$. Given a set of facts F, with say that P entails F, denoted $P \models F$, iff $P \models \alpha A$ for all $\alpha A \in F$. For ease, we will also represent an interpretation I as a set of ground facts $\{\alpha A : I \models \alpha A\}$. In particular, an interpretation may be seen as a pDatalog program.

2.2 Web directories and mappings

Web directories

A *web directory* is a pair $\langle \mathbf{C}, \preceq \rangle$, where $\mathbf{C} = \{C_1, ..., C_n\}$ is a finite non-empty set of *classes* (or concepts, or categories) and $\preceq$ is a partial order on $\mathbf{C}$ with a top class $\top$ (for all $C \in \mathbf{C}, C \preceq \top$). The intended meaning of $C_1 \preceq C_2$ is that the class C_1 is more specific than the class C_2, i.e. all instances of C_1 are instances of C_2 (see Fig. 1).

From a logical point of view, we assume that each class C_i is an unary predicate denoting the set of object identifiers of the instances of class C_i. For ease, in the remaining of this paper, we will always assume that the object identifiers belong to a set $\mathcal{X}$. Given a web directory $(\mathbf{C}, \preceq)$ and an interpretation I, we say that I is a model of $(\mathbf{C}, \preceq)$ iff $C_1^I \models C_2^I$ whenever $C_1 \preceq C_2$. Essentially, this says that each instance of C_1 is an instance of C_2 with the same probability. Given a web directory $(\mathbf{C}, \preceq)$ and a model I of it, then the *instantiation* of $(\mathbf{C}, \preceq)$ under I, denoted $(\mathbf{C}, \preceq)^I$, is the tuple $\mathbf{C}^I = \langle C_1^I, ..., C_n^I \rangle$, i.e the tuple of all class instantiations under I [5]. Of course, an object being an instance of a class C has also attributes (sometimes called properties). Each attribute A can be modelled as a predicate $A(x, v_1, \ldots, v_l)$ indicating that the value of the attribute A of the object identified with the object identifier x is $v_1, \ldots, v_l$. For the sake of our purpose, in this paper, we use one binary relation `content` only which stores the object identifiers and the text related to them, i.e. $\texttt{content}^I \subset \mathcal{X} \times \mathcal{T}$, where $\mathcal{T}$ is the string data type. This models scenarios of Web directories where the objects are web pages. Finally, note that a web directory may easily be encoded into pDatalog as a set of rules

$$C(x) \leftarrow C'(x) \ ,$$

for all $C' \preceq C$.

Web directory mappings

Our goal is to automatically determine "similarity" relationships between classes of two web directories. For instance, given the web directories in Fig. 1, we would like to determine that an instance of the class "Latin American History" in the Cornell Courses Catalogue is likely an instance of the "History of the Americas" in the Washington Courses Catalogue and that "History of the Americas" is the most specific class having this property (in order to prefer the former mapping onto the "Latin American History" $\mapsto$ "History" mapping).

Theoretically, web directory mappings follow the so-called GLaV approach [25]: a *mapping* is a tuple $\mathcal{M} = (\mathbf{T}, \mathbf{S}, \Sigma)$, where $\mathbf{T}$ denotes the target (global) web directory and $\mathbf{S}$ the source (local) web directory with no relation symbol in common, and Σ is a finite set of *mapping constraints* (pDatalog rules) of the form:

$$\alpha_{j,i} \ T_j(x) \leftarrow S_i(x) \ ,$$

where T_j and S_i are target and source classes, respectively, and x is a variable ranging over object identifiers. The intended meaning of the above rules is that the class S_i of the source web directory is mapped onto the class T_j of the target web directory and the probability that this mapping is indeed true is given by $\alpha_{j,i}$. Note that a source class may be mapped onto several target

[5] One might wonder why we consider the tuple $\langle C_1^I, ..., C_n^I \rangle$ rather than the set $\{C_1^I, ..., C_n^I\}$. The reason is that in the latter case two classes may collapse together, a behaviour we want to avoid.

classes and a target class may be the target of many source classes, i.e. we may have complex mappings

$$\Sigma \supseteq \{\alpha_{1,1}\ T_1(x) \leftarrow S_1(x), \alpha_{1,2}\ T_1(x) \leftarrow S_2(x), \alpha_{2,1}\ T_2(x) \leftarrow S_1(x)\}\ .$$

But, we do not require that we have a mapping for every target class.

For a web directory mapping $\mathcal{M} = (\mathbf{T}, \mathbf{S}, \Sigma)$ and a fixed model I for $\mathbf{S}$, a model J for $\mathbf{T}$ is a *solution* for I under $\mathcal{M}$ if and only if $\langle J, I\rangle$ (the combined interpretation over $\mathbf{T}$ and $\mathbf{S}$) is a model of Σ. The minimal solution is denoted by $J(I, \Sigma)$, the corresponding instance of $\mathbf{T}$ using interpretation $J(I, \Sigma)$ is denoted with $\mathbf{T}(I, \Sigma)$ (which is also called a minimal solution). Essentially, given a model I of $\mathbf{S}$, $\mathbf{T}(I, \Sigma)$ is the "translation/exchange" of the instances in the source web directory $\mathbf{S}^I$ into instances of the target web directory $\mathbf{T}$.

3 Learning web directory mappings

Learning a web directory mapping in oPLMap consists of four steps:

1. we guess a potential web directory mapping, i.e. a set of rules Σ_k of the form $T_j(x) \leftarrow S_i(x)$ (rules without weights yet);
2. we estimate the quality of the mapping Σ_k;
3. among all possible sets Σ_k, we select the "best" web directory mapping according to our quality measure; and finally
4. the weights α for rules in the selected web directory mapping have to be estimated.

3.1 Estimating the quality of a mapping

Consider a target web directory $\mathbf{T} = (\{T_1, \ldots, T_t\}, \preceq_\mathbf{T})$ and a source web directory $\mathbf{S} = (\{S_1, \ldots, S_s\}, \preceq_\mathbf{S})$, and two models I of $\mathbf{S}$ and J of $\mathbf{T}$. Consider I and its minimal solution $J(I, \Sigma)$ and the corresponding instance, $\mathbf{T}(I, \Sigma)$, of $\mathbf{T}$ using interpretation $J(I, \Sigma)$. Note that $\mathbf{T}(I, \Sigma)$ contains instances of classes in $\mathbf{T}$ and each instance has its own content. For instance (see Fig. 1), consider the mapping $\mathcal{M} = (\mathbf{T}, \mathbf{S}, \Sigma)$, with $\mathbf{T}$ and $\mathbf{S}$ containing the classes

$$\mathtt{T} = \mathtt{History\_of\_the\_\ Americas}$$

$$\mathtt{S} = \mathtt{Latin\_American\_History}$$

and consider the mapping

$$\Sigma \supseteq \{\mathtt{T(X)} \leftarrow \mathtt{S(X)}\}\ .$$

Suppose we have a model I of the source web directory $\mathbf{S}$ with two instances identified with `x1` and `x2` of the class `S`,

$$\mathtt{S}^I = \{\mathtt{S(x1)}, \mathtt{S(x2)}\} \ ,$$

where their content is

$$\begin{array}{l}\mathtt{content(x1,\text{“}A\ survey\ of\ Mexico's\ history...\text{”})} \ , \\ \mathtt{content(x2,\text{“}...questions\ of\ gender\ in\ Latin\ America...\text{”})} \ .\end{array}$$

Similarly, suppose we have a model J of the target web directory $\mathbf{T}$ with two instances identified with `x3` and `x4` of the class `T`,

$$\mathtt{T}^I = \{\mathtt{T(x3)}, \mathtt{T(x4)}\} \ ,$$

where their content is

$$\begin{array}{c}\mathtt{content(x3,\text{“}History\ of\ Latin\ America\ from\ colonial} \\ \mathtt{beginnings\ to\ the\ present...\text{”})} \ , \\ \mathtt{content(x4,\text{“}The\ American\ people\ and\ their\ culture\ in} \\ \mathtt{the\ modern\ era...\text{”})} \ .\end{array}$$

Then the minimal solution $J' = J(I, \Sigma)$ and the corresponding instance, $\mathbf{T}(I, \Sigma)$, of $\mathbf{T}$ is

$$\mathtt{T}^{J'} = \{\mathtt{T(x1)}, \mathtt{T(x2)}\} \ .$$

Note that the facts in $\mathtt{T}^J$ and $\mathtt{T}^{J'}$ differ in their identifiers, but there is some "semantic overlapping" according to their content.

Our goal is to find this semantic overlapping. In particular, our goal is to find the "best" set of mapping constraints Σ, which maximises the probability $Pr(\Sigma, J, I)$ that the objects in the minimal solution $\mathbf{T}(I, \Sigma)$ under $\mathcal{M} = (\mathbf{T}, \mathbf{S}, \Sigma)$ and the objects in $\mathbf{T}^J$ are similar.

Formally, consider the minimal solution $\mathbf{T}(I, \Sigma)$ and consider a class T_j of the target web directory. With $T_j(I, \Sigma)$ we denote the restriction of $\mathbf{T}(I, \Sigma)$ to the instance of the class T_j only. Then it can be verified that Σ can be *partitioned* into sets Σ_j, where each rule in Σ_j refers to the same target class T_j (all rules in Σ_j have T_j in the head), whose minimal solutions $T_j(I, \Sigma_j)$ only contain facts for T_j:

$$\begin{aligned} \Sigma_j &= \{r : r \in \Sigma, T_j \in head(r)\} \ , \\ \mathbf{T}(I, \Sigma) &= \cup_{j=1}^{t} T_j(I, \Sigma_j) \ , \\ \emptyset &= T_j(I, \Sigma_j) \cap T_k(I, \Sigma_k), \text{ if } j \neq k \ . \end{aligned}$$

Therefore, each target class can be considered independently:

$$Pr(\Sigma, J, I) = \prod_{j=1}^{t} Pr(\Sigma_j, J, I) \ .$$

We define $T_j(I, \Sigma_j)$ and T_j being similar iff $T_j(I, \Sigma_j)$ is similar to T_j and vice-versa. Thus, $Pr(\Sigma_j, J, I)$ can be computed as:

$$\begin{aligned} Pr(\Sigma_j, J, I) &= Pr(T_j|T_j(I, \Sigma_j)) \cdot Pr(T_j(I, \Sigma_j)|T_j) \\ &= Pr(T_j(I, \Sigma_j)|T_j)^2 \cdot \frac{Pr(T_j)}{Pr(T_j(I, \Sigma_j))} \\ &= Pr(T_j(I, \Sigma_j)|T_j)^2 \cdot \frac{|T_j|}{|T_j(I, \Sigma_j)|} \ . \end{aligned}$$

As building blocks of Σ_j, we use the sets $\Sigma_{j,i}$ containing just on rule:

$$\Sigma_{j,i} = \{\alpha_{j,i} \ T_j(x) \leftarrow S_i(x)\} \ . \tag{1}$$

For s source classes and a fixed j, there are also s possible sets $\Sigma_{j,i}$, and $2^s - 1$ non-empty combinations (unions) of them, forming all possible non-trivial sets Σ_j.

To simplify the notation, in the following we set $S_i = T_j(I, \Sigma_{j,i})$ for the instance derived by applying the single rule (1). For computational simplification, we assume that S_{i_1} and S_{i_2} are disjoint for $i_1 \neq i_2$. Then, for

$$\Sigma_j = \bigcup_{l=1}^{r} \Sigma_{j,i_l}$$

with indices $i_1, \ldots, i_r$, we obtain:

$$Pr(T_j(I, \Sigma_j)|T_j) = \sum_{l=1}^{r} Pr(S_{i_l}|T_j) \ . \tag{2}$$

Thus, to compute $Pr(\Sigma_j, J, I)$, we need to compute the $\mathcal{O}(s \cdot t)$ probabilities $Pr(S_i|T_j)$, which we will address in the next section.

3.2 Estimating the probability of a rule

Computing the quality of a mapping requires the probability $Pr(S_i|T_j)$, while the rule weight is $\alpha_{j,i} = Pr(T_j|S_i)$. This latter probability can easily computed from $Pr(S_i|T_j)$ as

$$Pr(T_j|S_i) = Pr(S_i|T_j) \cdot \frac{Pr(T_j)}{Pr(S_i)} = Pr(S_i|T_j) \cdot \frac{|T_j|}{|S_i|} \ . \tag{3}$$

Similar to GLUE [10, 11], the probability $Pr(S_i|T_j)$ is estimated by combining different classifiers $CL_1, \ldots CL_n$:

$$Pr(S_i|T_j) \approx Pr(S_i|T_j, CL_1, \ldots, CL_l) = \sum_{k=1}^{n} Pr(S_i|T_j, CL_k) \cdot Pr(CL_k) \ . \quad (4)$$

where the predictions $Pr(S_i|T_j, CL_k)$ is the estimate of the classifier CL_k for $Pr(S_i|T_j)$. By combining (2) and (4) we get

$$Pr(T_j(I, \Sigma_j)|T_j) = \sum_{k=1}^{n} Pr(CL_k) \cdot \sum_{l=1}^{r} Pr(S_{i_l}|T_j, CL_k) \ . \quad (5)$$

The probability $Pr(CL_k)$ describes the probability that we rely on the judgment of classifier CL_k, which can for example be expressed by the confidence we have in that classifier. We simply use $Pr(CL_k) = \frac{1}{n}$ for $1 \leq k \leq n$, i.e. the predictions are averaged.

In practice, each classifier CL_k computes a weight $w(S_i, T_j, CL_k)$, which is the classifier's initial approximation of $Pr(S_i|T_j)$. This weight $w(S_i, T_j, CL_k)$ will be then normalized and transformed into a probability

$$Pr(S_i|T_j, CL_k) = f(w(S_i, T_j, CL_k)) \ ,$$

the classifier's approximation of $Pr(S_i|T_j)$. All the probabilities $Pr(S_i|T_j, CL_k)$ will then be combined together as we will see later on. The normalization process is necessary as we combine the classifier estimates, which are heterogeneous in scale. Normalization is done in two steps. First, we can consider different normalization functions:

$$\begin{aligned} f_{id}(x) &= x \ , \\ f_{sum}(x) &= \frac{x}{\sum_{i'} w(S_{i'}, T_j, CL_k)} \ , \\ f_{lin}(x) &= c_0 + c_1 \cdot x \ , \\ f_{log}(x) &= \frac{\exp(b_0 + b_1 \cdot x)}{1 + \exp(b_0 + b_1 \cdot x)} \ . \end{aligned}$$

The functions f_{id}, f_{sum} and the logistic function f_{log} return values in $[0, 1]$. For the linear function, results below zero have to mapped onto zero, and results above one have to be mapped onto one. The function f_{sum} ensures that each value is in $[0, 1]$, and that the sum equals 1. Its biggest advantage is that is does not need parameters that have to be learned. In contrast, the parameters of the linear and logistic function are learned by regression in a system-training phase. This phase is only required once, and their results can be used for learning arbitrary many web directory mappings. Of course, normalization functions can be combined. In some cases it might be useful to bring the classifier weights in the same range (using f_{sum}), and then to apply another normalization function with parameters (e.g. the logistic function).

For the final probability $Pr(S_i|T_j, CL_k)$, we have the constraint

$$0 \leq Pr(S_i|T_j, CL_k) \leq \frac{\min(|S_i|, |T_j|)}{|T_j|} = \min(\frac{|S_i|}{|T_j|}, 1) \ .$$

Thus, the normalized value (which is in $[0, 1]$) is multiplied with $\min(|S_i|/|T_j|, 1)$ in a second normalization step.

It is worth noting that some of the classifiers consider the web directories only, while others are based on the textual content, i.e. the binary relation `content`, which associates a text with an object. The classifiers require instances of both web directories. However, these instances do not need to describe the same objects. Below, we describe the classifiers used in this paper.

Same class name stems

This binary classifier CL_S returns a weight of 1 if and only if the names of the two classes have the stem (using e.g. a Porter stemmer), and 0 otherwise:

$$w(S_i, T_j, CL_S) = \begin{cases} 1 \text{ if } S_i, T_j \text{ have same stem} \ , \\ 0 \text{ otherwise} \ . \end{cases}$$

Coordination-level match on class names

This classifier CL_{N-clm} employs information retrieval (IR) techniques by applying the coordination-level match similarity function onto the class name. For this, the class names S_i and T_j are considered as bags (multi-sets) of words; the words are obtained by converting the name into lower case, splitting it into tokens, removing stop words (frequent words without semantics), and apply stemming on the remaining tokens (which maps different derivations onto a common word "stem", e.g. "computer" and "computing" onto "comput"). The prediction is computed as overlap of the resulting sets of both class names:

$$w(S_i, T_j, CL_{N-clm}) = \frac{|S_i \cap T_j|}{|S_i \cup T_j|} \ .$$

Coordination-level match on class path names

This classifier CL_{PN-clm} is equivalent to CL_{N-clm}, but is applied on the complete "path" of a class C. With $C \preceq C_1 \preceq \cdots \preceq C_n \preceq \top$, this is the concatenation of the names of C as well as the names of $C_1, \ldots, C_n$. This concatenation is considered as a bag of words, and the same weights as for CL_{N-clm} are computed.

kNN classifier

A popular classifier for text and facts is kNN [35], which also employs IR techniques. For CL_{kNN}, each class S_i acts as a category, and training sets are formed from the instances of S_i:

$$Train = \bigcup_{i=1}^{s} \{(S_i, x, v) : (x, v) \in \texttt{content},\ x \in S_i\} \ .$$

For every instance $x \in T_j$ and its content v (i.e., the value v with $(x, v) \in$ $\texttt{content}$)[6], the k-nearest neighbours TOP_k have to be found by ranking the values $(S_i, x', v') \in Train$ according to their similarity $RSV(v, v')$. The prediction weights are then computed by summing up the similarity values for all x' which are built from S_i, and by averaging these weights $\tilde{w}(x, v, S_i)$ over all instances $x \in T_j$:

$$\begin{aligned}
w(S_i, T_j, CL_{kNN}) &= \frac{1}{|T_j|} \cdot \sum_{(x,v)\in\texttt{content},\ x\in T_j} \tilde{w}(x, v, S_i) \ , \\
\tilde{w}(x, v, S_i) &= \sum_{(S_l, x' v') \in TOP_k, S_i = S_l} RSV(v, v') \ , \\
RSV(v, v') &= \sum_{w \in v \cap v'} Pr(w|v) \cdot Pr(w|v') \ , \\
Pr(w|v) &= \frac{tf(w, v)}{\sum_{w' \in v} tf(w', v)} \ , \\
Pr(w|v') &= \frac{tf(w, v')}{\sum_{w' \in v'} tf(w', v')} \ .
\end{aligned}$$

Here, $tf(w, v)$ denotes the number of times the word w appears in the string v (seen as a bag of words). The quantity $tf(w, v')$ is similar.

Naive Bayes text classifier

The classifier CL_B uses a naive Bayes text classifier [35] for text content. As for the other classifiers, each class acts as a category, and class values are considered as bags of words (with normalized word frequencies as probability estimations). For each $(x, v) \in \texttt{content}$ with $x \in T_j$, the probability $Pr(S_i|v)$ that the value v should be mapped onto S_i is computed. In a second step, these probabilities are combined by:

[6] By abuse of notation, with $x \in T_j$ we denote that object x is an instance of class T_j, i.e. $T_j(x) \in T_j^J$. Thus, $(x, v) \in \texttt{content}$ is used as a shorthand for $\texttt{content}(x, v) \in \texttt{content}^J$.

$$w(S_i, T_j, CL_B) = \sum_{(x,v)\in\texttt{content},\ x\in T_j} Pr(S_i|v) \cdot Pr(v) \ .$$

Again, we consider the values as bags of words. With $Pr(S_i)$ we denote the probability that a randomly chosen value in $\bigcup_k S_k$ is a value in S_i, and $Pr(w|S_i) = Pr(w|v(S_i))$ is defined as for kNN, where $v(S_i) = \bigcup_{(x,v)\in\texttt{content},\ x\in S_i} v$ is the combination of all words in all values for all objects in S_i (again considered as bags). If we assume independence of the words in a value, then we obtain:

$$Pr(S_i|v) = Pr(v|S_i) \cdot \frac{Pr(S_i)}{Pr(v)} = \frac{Pr(S_i)}{Pr(v)} \cdot \prod_{w\in v} Pr(w|S_i) \ .$$

Together, the final formula is:

$$w(S_i, T_j, CL_B) = Pr(S_i) \cdot \sum_{(x,v)\in\texttt{content},\ x\in T_j} \prod_{w\in v} Pr(w|S_i) \ .$$

If a word does not appear in the content for any object in S_i, i.e. $Pr(w|S_i) = 0$, we assume a small value to avoid a product of zero.

3.3 Exploiting the hierarchical structure

So far, the oPLMap learning approach does not exploit the hierarchical nature of the web directories, i.e. the partial orders $\preceq_{\mathbf{T}}$ and $\preceq_{\mathbf{S}}$. To do so, we apply additional classifiers after we have computed the prediction $w'(S_i, T_j) = Pr(S_i|T_j, CL_1, \ldots, CL_l)$ from the so far considered classifiers $CL_1, \ldots, CL_l$. The rationale of this separation is that we want to avoid cyclic dependencies between hierarchical and non-hierarchical classifiers. The predictions of the hierarchical classifiers can then be combined with the predictions of the previous classifiers as before. Formally, we introduce the set B with the best matchings, i.e. where S_i is the best attribute on which T_j can be mapped onto: $B = \{(S_i, T_j)\colon w'(S_i, T_j) \geq \max_{S'} w'(S', T_j)\}$.

Matching parents

The binary classifier CL_P returns 1 if and only if two parents of the two source and target classes have highest matching prediction for all other $S_{i'}$ classes:

$$p_{\mathbf{S}}(C) = \{C'\colon C \preceq_{\mathbf{S}} C'\}$$
$$p_{\mathbf{T}}(C) = \{C'\colon C \preceq_{\mathbf{T}} C'\}$$

$$w(S_i, T_j, CL_P) = \begin{cases} 1 \text{ if } p_{\mathbf{S}}(S_i) \times p_{\mathbf{T}}(T_j) \cap B \neq \emptyset \ , \\ 0 \text{ otherwise} \ . \end{cases}$$

Matching children

The classifier CL_C returns the amount of matching children:

$$\begin{aligned} c_{\mathbf{S}}(C) &= \{C' : C' \preceq_{\mathbf{S}} C\} \\ c_{\mathbf{T}}(C) &= \{C' : C' \preceq_{\mathbf{T}} C\} \\ C(S_i, T_j) &= c_{\mathbf{S}}(S_i) \times c_{\mathbf{T}}(T_j) \\ w(S_i, T_j, CL_C) &= \frac{|C(S_i,T_j) \cap B|}{|C(S_i,T_j)|} \ . \end{aligned}$$

3.4 Additional constraints

Additional constraints can be applied on the learned rules for improving precision. These constraints are used after the sets of rules are learned for all target classes: we remove learned rules that violate one of these constraints. These constraints are stated against the hierarchical structure of the web directories:

1. We can assume that parent-child relationships in **S** and **T** are not reversed. In other words, we assume that for $S_{i_1} \preceq_{\mathbf{S}} S_{i_2}$ and $T_{j_1} \preceq_{\mathbf{T}} T_{j_2}$, it is not possible to map S_{i_1} onto T_{j_2} and S_{i_2} onto T_{j_1} together.
2. We can assume that if a source class S_{i_2} is parent of another source class S_{i_1}, then target classes onto which S_{i_2} is mapped are parents of target classes onto which S_{i_1} are mapped. Thus, $S_{i_1} \preceq_{\mathbf{S}} S_{i_2}$ and two rules $T_{j_1}(x) \leftarrow S_{i_1}(x)$ and $T_{j_2}(x) \leftarrow S_{i_2}(x)$ implies $T_{j_1} \preceq_{\mathbf{T}} T_{j_2}$.
3. Another assumption is that there is at most one rule for the target class. This will reduce the number of rules produced, and hopefully increase the percentage of correct rules.
4. We can drop all rules whose weight $\alpha_{j,i}$ is lower than a threshold ε, e.g. with $\varepsilon = 0.1$.
5. We can rank the rules according to their weights (in decreasing order), and use the n top-ranked rules (e.g. $n = 50$).

If a constraint is violated, the rule with the lower weight will be removed.

4 Experiments

This section describes the results from the oPLMap evaluation.

4.1 Evaluation setup

This section describes the test set (source and target instances) and the classifiers used for the experiments. It also introduces different effectiveness measurements for evaluating the learned web directory mappings (error, precision, recall). Experiments were performed on the "course catalog" test bed [7]. The Cornell University course catalog consists of 176 classes, among them 149 leaf concepts (in a maximum depth of 4), and 4,360 instances. The University of Washington contains 147 classes (141 leaf classes, maximum depth is again 4), and 6,957 instances.

Each collection is split randomly into four sub-collections of approximately the same size. The first sub-collection is always used for learning the parameters of the normalization functions (same documents in both web directories). The second sub-collection is used as source instance for learning the rules, and the third sub-collection is used as the target instance. Finally, the fourth sub-collection is employed for evaluating the learned rules (for both instances, i.e. we evaluate on parallel corpora). Rules are learned for both directions.

Each of web directory- and text-based classifiers introduced in section 3.2 are used alone, plus the combinations of all of these classifiers. For the hierarchical classifiers, none, both classifiers are used alone and in combination. In every experiment, every classifier used the same normalization function from section 3.2 and combinations of them.

$Pr(T_j(x) \in T_j^J)$ denotes the probability of a tuple x to be instance of the target attribute T_j in the model J of $\mathbf{T}$, i.e.: $Pr(T_j(x) \in T_j^J) = \alpha$ iff $T_j^J \models \alpha T_j(x)$ iff $\alpha T_j(x) \in T_j^J$. Often the target instance only contains deterministic data, then we have $Pr(T_j(x) \in T_j^J) \in \{0,1\}$. Similarly, $Pr(T_j(x) \in T_j(I,\Sigma_j)) \in [0,1]$ denotes the probability of a tuple x to be instance of the target attribute T_j in the minimal model $T_j(I,\Sigma_j)$ of the mapping $\mathcal{M} = (\mathbf{T},\mathbf{S},\Sigma)$ w. r. t. the model I of the source schema $\mathbf{S}$. Remind that $T_j(I,\Sigma_j)$ is obtained by applying all rules in Σ to the elements in the source instance $\mathbf{S}^I$ and then project the result on the target attribute T_j. Finally, the error of the mapping is defined by:

$$E(\mathcal{M}) = \frac{1}{\sum_j |U_j|} \sum_j \sum_{x \in U_j} (Pr(T_j(x) \in T_j^J) - Pr(T_j(x) \in T_j(I,\Sigma_j)))^2 ,$$

where $U_j = \{T_j(x) \in T_j^J\} \cup \{T_j(x) \in T_j(I,\Sigma_j)\}$. Furthermore, we evaluated if the learning approach computes the correct rules (neglecting the corresponding rule weights). Similar to the area of Information Retrieval [2], precision defines how many learned rules are correct, and how many correct rules are learned. In the following, R_L denotes the set of rules (without weights) returned by the learning algorithm, and R_A the set of rules (again without weights), which are the actual ones.

As we deal with hierarchical web directories, this hierarchy should also be included in the measures. Thus, for $S_1 \preceq_{\mathbf{S}} S_2$ and the rules

$$R_L = \{T_1(x) \leftarrow S_1(x)\}$$
$$R_A = \{T_1(x) \leftarrow S_2(x)\}$$

[7] `http://anhai.cs.uiuc.edu/archive/domains/course_catalog.html`

traditional precision and recall would be zero with these definitions:

$$precision_{trad} = \frac{|R_L \cap R_A|}{|R_L|} \ , \quad recall_{trad} = \frac{|R_L \cap R_A|}{|R_A|} \ .$$

However, the learned rule is too specific, but not completely wrong. The following definition takes that into consideration. With $d(C_1, C_2)$, we denote the distance between two classes C_1 and C_2 (from the same web directory) in the hierarchy:

$$d(C_1, C_2) = \begin{cases} 0 \ \text{ if } C_1 = C_2 \ , \\ n \ \text{ if } C_1 = C_{i_0} \preceq \ldots \preceq C_{i_n} = C_2 \ , C_{i_j} \neq C_{i_k} \text{ for } j \neq k \ , \\ n \ \text{ if } C_2 = C_{i_0} \preceq \ldots \preceq C_{i_n} = C_1 \ , C_{i_j} \neq C_{i_k} \text{ for } j \neq k \ , \\ \infty \text{ otherwise } \ . \end{cases}$$

For a mapping rule $r = T_j(x) \leftarrow S_i(x)$, $h(r) = T_j$ denotes the target class and $b(r) = S_i$ the source class. Then, these similarity measures are defined:

$$sim(r, r') = \begin{cases} \frac{1}{1+d(b(r), b(r'))} & \text{if } h(r) = h(r') \ , \\ 0 & \text{otherwise } \ , \end{cases}$$

$$sim(r, R) = \min_{r' \in R, sim(r, r') > 0} sim(r, r') \ .$$

These similarities are employed in the definition of hierarchy-based precision and recall:

$$precision = \frac{\sum_{r \in R_L} sim(r, R_A)}{|R_L|} \ , \quad recall = \frac{\sum_{r \in R_A} sim(r, R_L)}{|R_A|} \ .$$

Precision measures the average distance of learned rules with the actual ones, while recall measures the average distance of actual rules with the learned ones. Traditional precision and recall are defined in a analogous way, but with equality as similarity measure. In addition, we also combine precision and recall in the F-measure:

$$F = \frac{2}{\frac{1}{precision} + \frac{1}{recall}} \ .$$

Finally, we also used a variant of traditional precision where we drop all rules for target classes for which there are no relationships at all. This measure shows how good our approach is when we only consider the target classes for which we can be successful.

4.2 Results

In the experiments presented in this section, the learning steps are as follows:

1. Find the best web directory mapping:
   a) Estimate the probabilities $Pr(S_i|T_j, CL_1, \ldots, CL_l)$ for every $S_i \in \mathbf{S}$, $T_j \in \mathbf{T}$ using the web directory-based and text-based classifiers;
   b) Estimate the probabilities $Pr(S_i|T_j)$ for every $S_i \in \mathbf{S}$, $T_j \in \mathbf{T}$ using all classifiers (if any hierarchical classifier is used);
   c) For every target relation T_j and for every non-empty subset of web directory mapping rules having T_j as head, estimate the probability $Pr(\Sigma_j, J, I)$;
   d) Select the rule set Σ_j, which maximizes the probability $Pr(\Sigma_j, J, I)$.
2. Estimate the weights $Pr(T_j|S_i)$ for the learned rules by converting $Pr(S_i|T_j)$, using (3).
3. Compute the error, precision and recall as described above.

The name-based classifiers are slightly modified, as the class names contain some identifier suffix like `Chinese_CHIN_27` (Washington) or `Chinese_30` (Cornell). These suffixes are removed before these classifiers are applied.

The results are depicted in tables 1–14. Note that they are averaged over both mapping directions Cornell to Washington and vice-versa.

Runs without constraints

Here, CL_{PN-clm} minimizes the error (0.1305, averaged over all hierarchical classifiers and normalization functions and both mapping directions), the combination of all classifiers performs about 8% worse. The error of the two content-oriented classifiers–kNN and Naive Bayes–is about 30% worse compared to CL_{PN-clm}, and Naive Bayes is slightly better than kNN. Precision in general is quite low, the highest value is obtained for CL_S (0.2553 on average). The low precision is due to the fact that the system generates a huge number of rules (sometimes several hundreds), while the are only about 50 valid mappings. The best recall is achieved by the combination of all non-hierarchical classifiers with 0.9177 on average. The content-oriented classifiers perform worst, kNN has recall of 0.2772 (nearly 70% worse), while Naive Bayes yields a recall of 0.1920 (about 80% worse). The hierarchical precision and recall values are only slightly better than the traditional ones.

Error and precision is optimized by the f_{sum} normalization function (average error is 0.1524, average precision is 0.0965). The error of the identity function is only slightly worse (1.5%). In addition, this function yields the best recall (0.6322 on average), followed by f_{sum}. Thus, learning parameters for the linear and logistic mapping function does not help.

Together, the combination of all non-hierarchical classifiers with the identity normalization function yields the lowest error (0.1145 on average), followed by CL_{PN-clm} (about 7% worse). Precision is optimized by using CL_S

Table 1. Error

	f_{id}	f_{sum}	$f_{lin} \circ f_{sum}$	$f_{log} \circ f_{sum}$
CL_S	0.2340	0.2439	0.2980	0.3025
CL_{N-clm}	0.2301	0.2932	0.3136	0.2934
CL_{PN-clm}	0.1786	0.1503	0.1515	0.1931
CL_{kNN}	0.2396	0.1971	0.2214	0.2751
CL_B	0.2644	0.1706	0.2252	0.2238
all	0.1139	0.1272	0.1609	0.1611
CL_S / CL_P	0.0778	0.0798	0.0893	0.0892
CL_{N-clm} / CL_P	0.1221	0.1582	0.1616	0.1602
CL_{PN-clm} / CL_P	0.0934	0.1274	0.1211	0.1070
CL_{kNN} / CL_P	0.1098	0.0946	0.1035	0.0943
CL_B / CL_P	0.1167	0.0864	0.1160	0.1190
all / CL_P	0.1087	0.1317	0.1655	0.1443
CL_S / CL_C	0.1554	0.1592	0.1843	0.1864
CL_{N-clm} / CL_C	0.2201	0.2512	0.2333	0.2073
CL_{PN-clm} / CL_C	0.1192	0.1236	0.1228	0.1349
CL_{kNN} / CL_C	0.2332	0.2298	0.2213	0.2791
CL_B / CL_C	0.2771	0.1622	0.2341	0.2413
all / CL_C	0.1195	0.1341	0.1580	0.1622
CL_S / CL_P+CL_C	0.1012	0.1019	0.1109	0.1110
CL_{N-clm} / CL_P+CL_C	0.1472	0.1669	0.1608	0.1528
CL_{PN-clm} / CL_P+CL_C	0.0971	0.1305	0.1246	0.1129
CL_{kNN} / CL_P+CL_C	0.1160	0.1069	0.1106	0.1003
CL_B / CL_P+CL_C	0.1230	0.0927	0.1221	0.1224
all / CL_P+CL_C	0.1159	0.1393	0.1669	0.1480

with any normalization function (virtually the same precision), followed by CL_{N-clm} with f_{sum} or f_{id} (70% worse). Finally, CL_{N-clm} with f_{id} or f_{sum} yields highest recall (9520 and 0.9405, respectively), follows by the classifier combination with the same normalization functions (about 2% worse).

The hierarchical classifier CL_P minimizes error with 0.1157, the combination with CL_C is 7.4% worse. CL_C alone performs more than 60% worse, and using no hierarchical classifier at all increases error by 90% compared to CL_P. This order is reversed for precision and (nearly) recall; best precision is obtained when no hierarchical classifier is used.

Average traditional precision is 0.0919. When only target classes which there are mappings are considered, then precision increases (0.2280 on average).

Runs with constraints

In general, precision is quite high, as the number of rules is pruned dramatically. Recall is lower when a constraint is used, but there are often cases where

Table 2. Traditional precision

	f_{id}	f_{sum}	$f_{lin} \circ f_{sum}$	$f_{log} \circ f_{sum}$
CL_S	0.6909	0.6909	0.6909	0.6909
CL_{N-clm}	0.0875	0.0874	0.0875	0.0779
CL_{PN-clm}	0.0339	0.0339	0.0339	0.0541
CL_{kNN}	0.1479	0.1479	0.1479	0.0714
CL_B	0.1525	0.1525	0.1525	0.0939
all	0.0512	0.0685	0.0688	0.0643
CL_S / CL_P	0.0527	0.0525	0.0516	0.0515
CL_{N-clm} / CL_P	0.0680	0.0684	0.0626	0.0549
CL_{PN-clm} / CL_P	0.0360	0.0176	0.0168	0.0088
CL_{kNN} / CL_P	0.0132	0.0191	0.0190	0.0159
CL_B / CL_P	0.0091	0.0127	0.0120	0.0116
all / CL_P	0.0522	0.0740	0.0725	0.0623
CL_S / CL_C	0.2249	0.2255	0.2255	0.2255
CL_{N-clm} / CL_C	0.0912	0.0958	0.0921	0.0815
CL_{PN-clm} / CL_C	0.0356	0.0314	0.0314	0.0448
CL_{kNN} / CL_C	0.0964	0.1235	0.1062	0.0741
CL_B / CL_C	0.0529	0.0993	0.1026	0.0790
all / CL_C	0.0514	0.0686	0.0692	0.0648
CL_S / CL_P+CL_C	0.0535	0.0531	0.0525	0.0526
CL_{N-clm} / CL_P+CL_C	0.0691	0.0698	0.0633	0.0545
CL_{PN-clm} / CL_P+CL_C	0.0368	0.0170	0.0164	0.0098
CL_{kNN} / CL_P+CL_C	0.0152	0.0195	0.0191	0.0165
CL_B / CL_P+CL_C	0.0105	0.0141	0.0137	0.0128
all / CL_P+CL_C	0.0525	0.0741	0.0728	0.0630

both precision and recall is sufficiently high. However, error is much higher; here, missing rules count much higher than wrong rules with a low weight.

For constraint 1, the differences in the order of the classifiers, normalization functions, the combination of classifier and normalization functions and the order of the hierarchical classifiers are small. In the following, we only present differences to the run without any constraint.

For constraint 2, the differences in the order of the classifiers is neglectable. In contrast, using a linear or logistic normalization function yields better error and precision than the other normalization functions.

The situation is different for constraint 3 (“at most one rule per target rule”), constraint 4 (“only rules with weight above 0.1”) and constraint 5 (“at most 40 rules”). Here, CL_S yields the best error and precision; CL_{PN-clm} performs quite bad.

Table 3. Traditional recall

	f_{id}	f_{sum}	$f_{lin} \circ f_{sum}$	$f_{log} \circ f_{sum}$
CL_S	0.6448	0.6448	0.6448	0.6448
CL_{N-clm}	0.9330	0.9330	0.9330	0.9230
CL_{PN-clm}	0.7730	0.7730	0.7730	0.8459
CL_{kNN}	0.2689	0.2689	0.2689	0.2396
CL_B	0.1241	0.1241	0.1241	0.2196
all	0.9315	0.9415	0.9415	0.8930
CL_S / CL_P	0.7119	0.7119	0.7119	0.7119
CL_{N-clm} / CL_P	0.9615	0.9337	0.8874	0.7841
CL_{PN-clm} / CL_P	0.7922	0.3533	0.3433	0.1633
CL_{kNN} / CL_P	0.2319	0.3152	0.3152	0.2774
CL_B / CL_P	0.1533	0.2004	0.1911	0.1919
all / CL_P	0.9315	0.9322	0.9137	0.8374
CL_S / CL_C	0.7119	0.7119	0.7119	0.7119
CL_{N-clm} / CL_C	0.9522	0.9615	0.9515	0.9322
CL_{PN-clm} / CL_C	0.8015	0.7052	0.7052	0.7681
CL_{kNN} / CL_C	0.2774	0.2689	0.2681	0.2489
CL_B / CL_C	0.1826	0.2096	0.2296	0.2396
all / CL_C	0.9315	0.9415	0.9515	0.9022
CL_S / CL_P+CL_C	0.7219	0.7219	0.7219	0.7219
CL_{N-clm} / CL_P+CL_C	0.9615	0.9337	0.8867	0.7826
CL_{PN-clm} / CL_P+CL_C	0.8015	0.3326	0.3226	0.1833
CL_{kNN} / CL_P+CL_C	0.2596	0.3252	0.3152	0.2867
CL_B / CL_P+CL_C	0.1833	0.2296	0.2311	0.2219
all / CL_P+CL_C	0.9315	0.9322	0.9237	0.8467

Comparison of constraints

The best error is obtained when no constraint is used at all (0.1622 on average), followed by constraint 1. All other constraints are more than 90% worse; the worst result is obtained when applying all constraints (0.7395). Similarly, recall is maximized when no constraint is used (0.5982 on average), followed by constraint 1. Again, applying all constraints yields the worst recall (0.2906). In contrast, the combination of all constraints yields the highest precision with 0.7584.

The F-measure combines precision and recall. Here, constraint 5 is the best with 0.3745 (where both precision and recall have about the same value), directly followed by the combination of all constraints.

The values for the hierarchical variants are higher (e.g. highest precision with 0.8227), but the order is nearly the same.

The highest hierarchical F-measure 0.6912 (precision of 0.7275, recall of 0.6356) is obtained for CL_S with any of the four normalization functions, without hierarchical classifier, and with constraint 4. The combination of all

Table 4. Traditional precision, Constraint 3

	f_{id}	f_{sum}	$f_{lin} \circ f_{sum}$	$f_{log} \circ f_{sum}$
CL_S	0.7125	0.7125	0.7008	0.7008
CL_{N-clm}	0.2716	0.2716	0.2716	0.2563
CL_{PN-clm}	0.1283	0.1283	0.1283	0.1443
CL_{kNN}	0.4330	0.4330	0.4330	0.2094
CL_B	0.1681	0.1873	0.1873	0.2265
all	0.2271	0.2352	0.2383	0.2293
CL_S / CL_P	0.2038	0.2038	0.2038	0.2038
CL_{N-clm} / CL_P	0.2205	0.2112	0.2019	0.1863
CL_{PN-clm} / CL_P	0.1365	0.0800	0.0769	0.0241
CL_{kNN} / CL_P	0.0415	0.0484	0.0449	0.0487
CL_B / CL_P	0.0159	0.0399	0.0330	0.0290
all / CL_P	0.2269	0.2322	0.2236	0.2264
CL_S / CL_C	0.5212	0.5125	0.5037	0.5037
CL_{N-clm} / CL_C	0.2869	0.2908	0.2944	0.2942
CL_{PN-clm} / CL_C	0.1394	0.1151	0.1180	0.1277
CL_{kNN} / CL_C	0.4330	0.4523	0.4330	0.2094
CL_B / CL_C	0.0769	0.1695	0.2265	0.2849
all / CL_C	0.2240	0.2414	0.2443	0.2322
CL_S / CL_P+CL_C	0.2069	0.2069	0.2069	0.2069
CL_{N-clm} / CL_P+CL_C	0.2360	0.2267	0.2112	0.1925
CL_{PN-clm} / CL_P+CL_C	0.1423	0.0707	0.0707	0.0272
CL_{kNN} / CL_P+CL_C	0.0415	0.0515	0.0415	0.0487
CL_B / CL_P+CL_C	0.0222	0.0396	0.0321	0.0415
all / CL_P+CL_C	0.2300	0.2355	0.2267	0.2293

classifiers with all constraints yield a slightly worse quality for CL_C and the $f_{log} \circ f_{sum}$ normalization function.

Example of learned rules

This rule has been learned using the kNN classifier:

```
0.1062 Mechanical_and_Aerospace_Engineering_143(X) :-
         Aeronautics_and_Astronautics_A_A_133(X).
```

Actually, this rule described the only mapping onto the target class `Mechanical_and_Aerospace_Engineering_143`:

```
Mechanical_and_Aerospace_Engineering_143(O) :-
  Mechanical_Engineering_145(O).
```

Table 5. Traditional recall, Constraint 3

	f_{id}	f_{sum}	$f_{lin} \circ f_{sum}$	$f_{log} \circ f_{sum}$
CL_S	0.5978	0.5978	0.5885	0.5885
CL_{N-clm}	0.6856	0.6856	0.6856	0.6470
CL_{PN-clm}	0.4081	0.4081	0.4081	0.4596
CL_{kNN}	0.2204	0.2204	0.2204	0.1078
CL_B	0.0848	0.0948	0.0948	0.1156
all	0.7263	0.7511	0.7611	0.7326
CL_S / CL_P	0.6078	0.6078	0.6078	0.6078
CL_{N-clm} / CL_P	0.6856	0.6578	0.6307	0.5830
CL_{PN-clm} / CL_P	0.4367	0.2570	0.2470	0.0770
CL_{kNN} / CL_P	0.1278	0.1463	0.1370	0.1456
CL_B / CL_P	0.0493	0.1141	0.0956	0.0878
all / CL_P	0.7256	0.7419	0.7148	0.7233
CL_S / CL_C	0.5978	0.5885	0.5793	0.5793
CL_{N-clm} / CL_C	0.7348	0.7448	0.7541	0.7533
CL_{PN-clm} / CL_C	0.4459	0.3681	0.3774	0.4089
CL_{kNN} / CL_C	0.2204	0.2304	0.2204	0.1078
CL_B / CL_C	0.0400	0.0863	0.1156	0.1463
all / CL_C	0.7163	0.7711	0.7804	0.7419
CL_S / CL_P+CL_C	0.6178	0.6178	0.6178	0.6178
CL_{N-clm} / CL_P+CL_C	0.7348	0.7078	0.6607	0.6030
CL_{PN-clm} / CL_P+CL_C	0.4552	0.2270	0.2270	0.0870
CL_{kNN} / CL_P+CL_C	0.1278	0.1563	0.1278	0.1456
CL_B / CL_P+CL_C	0.0693	0.1148	0.0978	0.1278
all / CL_P+CL_C	0.7356	0.7526	0.7248	0.7326

5 Conclusion, Related work and outlook

With the proliferation of data sharing applications over the Web, involving ontologies (and in particular web directories), the development of automated tools for ontology matching will be of particular importance. In this paper, we have presented a Probabilistic, Logic-based formal framework (*oPLMap*) for ontology Matching involving web directories. The peculiarity of our approach is that it combines neatly machine learning and heuristic techniques, for learning a set of mapping rules, with logic, in particular probabilistic Datalog. This latter aspect is of particular importance as it constitutes the basis to extend our "ontological "model to more expressive formal languages for ontology description, like OWL DL [21] in particular (founded on so-called *Description Logics* [1]), which are the state of the art in the Semantic Web. As a consequence, all aspects of logical reasoning, considered as important in the Semantic Web community [8], can easily be plugged into our model. Our

[8] http://www.semanticweb.org/

Table 6. Traditional precision, Constraint 4

	f_{id}	f_{sum}	$f_{lin} \circ f_{sum}$	$f_{log} \circ f_{sum}$
CL_S	0.7063	0.7215	0.7275	0.7275
CL_{N-clm}	0.1052	0.2519	0.4184	0.5342
CL_{PN-clm}	0.0407	0.0490	0.2500	0.0000
CL_{kNN}	0.0000	0.5441	0.4097	0.0848
CL_B	0.0000	0.2833	0.2664	0.1667
all	0.1502	0.5969	0.6892	0.7033
CL_S / CL_P	0.7158	0.7386	0.7326	0.7326
CL_{N-clm} / CL_P	0.1500	0.4319	0.6208	0.6511
CL_{PN-clm} / CL_P	0.0520	0.1623	0.5625	0.6250
CL_{kNN} / CL_P	1.0000	0.5000	1.0000	0.0884
CL_B / CL_P	1.0000	0.3182	0.3500	0.7500
all / CL_P	0.2045	0.6409	0.7766	0.7710
CL_S / CL_C	0.7074	0.7367	0.7305	0.7305
CL_{N-clm} / CL_C	0.1415	0.4342	0.5741	0.6286
CL_{PN-clm} / CL_C	0.0512	0.0833	0.5000	0.1250
CL_{kNN} / CL_C	0.7500	0.4929	0.5000	0.0836
CL_B / CL_C	0.5000	0.2500	0.3167	0.5000
all / CL_C	0.2102	0.6307	0.7636	0.7571
CL_S / CL_P+CL_C	0.7136	0.7305	0.7514	0.7403
CL_{N-clm} / CL_P+CL_C	0.2560	0.5264	0.6186	0.6519
CL_{PN-clm} / CL_P+CL_C	0.0697	0.5833	0.7500	0.4500
CL_{kNN} / CL_P+CL_C	0.7500	0.6667	0.7500	0.6250
CL_B / CL_P+CL_C	0.7500	0.3071	0.4167	0.7500
all / CL_P+CL_C	0.2850	0.6458	0.8001	0.7555

logical foundation also eases the formalization of the so-called *query reformulation task* [25], which tackles the issue of converting a query over the target ontology into one (or more) queries over the source ontology.

Our model oPLMap has its foundations in three strictly related research areas: *schema matching* [34], *information integration* [25] and *information exchange* [14], in particular, and borrows from them the terminology and ideas. Indeed, related to the latter, we view the matching problem as the problem of determining the "best possible set Σ of formulae of a certain kind" such that the *exchange* of instances of a source class into a target class has highest probability of being correct. From information integration we inherit the type of rules we are looking for. Indeed, we have a so-called GLaV model [25]. From the former we inherit the requirement to rely on machine learning techniques to automate the process of schema matching. Additionally, a side effect is that we can inherit many of the theoretical results developed in these areas so far, especially from the latter two (see, e.g., [4, 14, 15, 16, 25, 29]).

Table 7. Traditional recall, Constraint 4

	f_{id}	f_{sum}	$f_{lin} \circ f_{sum}$	$f_{log} \circ f_{sum}$
CL_S	0.6448	0.6448	0.6356	0.6356
CL_{N-clm}	0.9237	0.7441	0.5507	0.4352
CL_{PN-clm}	0.7452	0.0193	0.0100	0.0100
CL_{kNN}	0.0200	0.2019	0.0678	0.2396
CL_B	0.0000	0.1241	0.0578	0.0093
all	0.8381	0.6578	0.5152	0.5444
CL_S / CL_P	0.6263	0.6170	0.5985	0.5985
CL_{N-clm} / CL_P	0.8774	0.5793	0.4274	0.3681
CL_{PN-clm} / CL_P	0.6989	0.0385	0.0193	0.0293
CL_{kNN} / CL_P	0.0385	0.0778	0.0478	0.1463
CL_B / CL_P	0.0193	0.1041	0.0293	0.0193
all / CL_P	0.8004	0.6378	0.4774	0.4574
CL_S / CL_C	0.6263	0.6170	0.5985	0.5985
CL_{N-clm} / CL_C	0.8681	0.5800	0.3804	0.3596
CL_{PN-clm} / CL_C	0.6896	0.0200	0.0100	0.0200
CL_{kNN} / CL_C	0.0193	0.0885	0.0193	0.1563
CL_B / CL_C	0.0100	0.0763	0.0393	0.0100
all / CL_C	0.8004	0.6193	0.4681	0.4674
CL_S / CL_P+CL_C	0.5985	0.5985	0.5615	0.5330
CL_{N-clm} / CL_P+CL_C	0.7533	0.4930	0.3041	0.3011
CL_{PN-clm} / CL_P+CL_C	0.6619	0.0293	0.0193	0.0293
CL_{kNN} / CL_P+CL_C	0.0193	0.0585	0.0193	0.0478
CL_B / CL_P+CL_C	0.0193	0.0670	0.0193	0.0193
all / CL_P+CL_C	0.7819	0.5722	0.4389	0.4381

The matching problem for ontologies, as well as the matching problem for schemas has been addressed by many researchers so far and are strictly related, as e.g. schemas can be seen as ontologies with restricted relationship types. The techniques applied in schema matching can be applied to ontology matching as well. Additionally, we have to take care of the hierarchies.

Related to ontology matching are, for instance, the works [10, 22, 24, 32] (see [10] for a more extensive comparison). While most of them use a variety of heuristics to match ontology elements, very few do use machine learning and exploit information in the data instances [10, 22, 24]. [24] computes the similarity between two concepts, as the similarity among the vector representations of the concepts (using Information Retrieval statistics like tf.idf). HICAL [22] uses κ-statistics from the data instances to infer rule mappings among concepts. Finally, [10, 11] (GLUE), but see also [6], is the most involved system. GLUE is based on the ideas introduced earlier by LSD [9]. Similar to our approach, it employed a linear combination of the predictions of multiple base learners (classifiers). The combination weights are learned via

Table 8. Traditional precision, Constraint 5

	f_{id}	f_{sum}	$f_{lin} \circ f_{sum}$	$f_{log} \circ f_{sum}$
CL_S	0.6909	0.6909	0.6909	0.6909
CL_{N-clm}	0.5600	0.4800	0.4800	0.4800
CL_{PN-clm}	0.3300	0.0400	0.0400	0.0600
CL_{kNN}	0.2700	0.2700	0.2700	0.1100
CL_B	0.1525	0.1525	0.1525	0.1700
all	0.5700	0.6300	0.6600	0.6700
CL_S / CL_P	0.6500	0.6500	0.6400	0.6400
CL_{N-clm} / CL_P	0.5800	0.5100	0.4800	0.4700
CL_{PN-clm} / CL_P	0.3400	0.0800	0.0800	0.0800
CL_{kNN} / CL_P	0.1500	0.1800	0.1600	0.1300
CL_B / CL_P	0.0500	0.1500	0.1100	0.0900
all / CL_P	0.5900	0.6400	0.6700	0.6700
CL_S / CL_C	0.6700	0.6700	0.6700	0.6700
CL_{N-clm} / CL_C	0.5700	0.4800	0.5000	0.5500
CL_{PN-clm} / CL_C	0.3200	0.1400	0.1600	0.1100
CL_{kNN} / CL_C	0.2500	0.2700	0.2800	0.1400
CL_B / CL_C	0.1000	0.1800	0.1800	0.1400
all / CL_C	0.5700	0.6400	0.6700	0.6800
CL_S / CL_P+CL_C	0.6500	0.6400	0.6300	0.6300
CL_{N-clm} / CL_P+CL_C	0.5900	0.5100	0.4800	0.4800
CL_{PN-clm} / CL_P+CL_C	0.3500	0.1200	0.1300	0.1000
CL_{kNN} / CL_P+CL_C	0.1500	0.1700	0.1500	0.1400
CL_B / CL_P+CL_C	0.0800	0.1600	0.1200	0.1100
all / CL_P+CL_C	0.5900	0.6600	0.6700	0.6600

regression on manually specified mappings between a small number of learning ontologies.

Related to schema matching are, for instance, the works [3, 6, 7, 8, 9, 13, 14, 18, 19, 23, 27, 28, 30, 33, 36] (see [34] for a more extensive comparison). As pointed out above, closest to our approach is [14] based on a logical framework for data exchange, but we incorporated the inherent uncertainty of rule mappings and classifier combinations (like LSD) into our framework as well. While the majority of the approaches focuses on finding 1-1 matchings (e.g. iMap [6] is an exception), we allow complex mappings and domain knowledge as well.

As future work, we see some appealing points. The combination of a rule-based language with an expressive ontology language has attracted the attention of many researchers (see, e.g., [12, 20] to cite a few) and is considered as an important requirement. Currently we are combining probabilistic Datalog with OWL DL so that complex ontologies can be described (so far, none of the approaches above addresses the issue of uncertainty). Besides this, as then

Table 9. Traditional recall, Constraint 5

	f_{id}	f_{sum}	$f_{lin} \circ f_{sum}$	$f_{log} \circ f_{sum}$
CL_S	0.6448	0.6448	0.6448	0.6448
CL_{N-clm}	0.5430	0.4637	0.4637	0.4630
CL_{PN-clm}	0.3204	0.0385	0.0385	0.0578
CL_{kNN}	0.2589	0.2589	0.2589	0.1078
CL_B	0.1241	0.1241	0.1241	0.1626
all	0.5522	0.6078	0.6370	0.6463
CL_S / CL_P	0.6263	0.6263	0.6170	0.6170
CL_{N-clm} / CL_P	0.5615	0.4930	0.4652	0.4552
CL_{PN-clm} / CL_P	0.3304	0.0770	0.0770	0.0770
CL_{kNN} / CL_P	0.1478	0.1756	0.1570	0.1263
CL_B / CL_P	0.0493	0.1433	0.1063	0.0878
all / CL_P	0.5715	0.6170	0.6463	0.6463
CL_S / CL_C	0.6448	0.6448	0.6448	0.6448
CL_{N-clm} / CL_C	0.5530	0.4637	0.4837	0.5300
CL_{PN-clm} / CL_C	0.3111	0.1356	0.1556	0.1070
CL_{kNN} / CL_C	0.2404	0.2589	0.2681	0.1370
CL_B / CL_C	0.0978	0.1726	0.1726	0.1356
all / CL_C	0.5522	0.6178	0.6463	0.6556
CL_S / CL_P+CL_C	0.6263	0.6170	0.6078	0.6078
CL_{N-clm} / CL_P+CL_C	0.5715	0.4930	0.4659	0.4659
CL_{PN-clm} / CL_P+CL_C	0.3396	0.1170	0.1270	0.0970
CL_{kNN} / CL_P+CL_C	0.1478	0.1663	0.1478	0.1363
CL_B / CL_P+CL_C	0.0793	0.1533	0.1178	0.1078
all / CL_P+CL_C	0.5715	0.6363	0.6463	0.6370

the instances of a class may be structured, e.g. have several attributes, or may be semi-structured (e.g. XML documents) we have to combine our ontology matching method with a so-called schema matching method (see, e.g. [9, 34]). We plan to integrate our model with the method based on [31] for schema matching, as the latter is rooted on the same principles of the work we have presented here. In particular, we are investigating several methods to learn mappings in an environment with ontologies and structured or semi-structured data: (*i*) to learn the schema mappings for each ontology class first, and ontology mappings in a second step; or (*ii*) both learning steps are performed simultaneously, which means that the quality of every possible mapping rule is estimated, and an overall optimum mapping subset is selected.

Additional areas of intervention rely on augmenting the effectiveness of the machine learning part. While to fit new classifiers into our model is straightforward theoretically, practically finding out the most appropriate one or a combination of them is quite more difficult, as our results show. In the future, more variants should be developed and evaluated to improve the quality of

Table 10. Traditional precision, All constraints

	f_{id}	f_{sum}	$f_{lin} \circ f_{sum}$	$f_{log} \circ f_{sum}$
CL_S	0.7790	0.7790	0.7790	0.7596
CL_{N-clm}	0.5600	0.4900	0.4928	0.5546
CL_{PN-clm}	0.3400	0.0714	0.2500	0.0000
CL_{kNN}	0.0000	0.4722	0.3250	0.1664
CL_B	0.0000	0.2798	0.2976	0.1667
all	0.6100	0.6625	0.7393	0.7421
CL_S / CL_P	0.7625	0.7732	0.7867	0.7663
CL_{N-clm} / CL_P	0.5800	0.5100	0.6934	0.7054
CL_{PN-clm} / CL_P	0.3400	0.2917	0.6250	0.7500
CL_{kNN} / CL_P	1.0000	0.4250	1.0000	0.2929
CL_B / CL_P	1.0000	0.3289	0.3500	0.7500
all / CL_P	0.6200	0.6951	0.7999	0.7952
CL_S / CL_C	0.7829	0.7936	0.7715	0.7715
CL_{N-clm} / CL_C	0.5700	0.4841	0.5872	0.6148
CL_{PN-clm} / CL_C	0.3800	0.0714	0.5000	0.0833
CL_{kNN} / CL_C	0.5000	0.5000	0.2500	0.1692
CL_B / CL_C	0.5000	0.2938	0.1500	0.5000
all / CL_C	0.6000	0.6970	0.7999	0.7837
CL_S / CL_P+CL_C	0.7887	0.7998	0.8117	0.8099
CL_{N-clm} / CL_P+CL_C	0.5900	0.5881	0.6952	0.7396
CL_{PN-clm} / CL_P+CL_C	0.4000	1.0000	1.0000	1.0000
CL_{kNN} / CL_P+CL_C	1.0000	0.7500	1.0000	0.7500
CL_B / CL_P+CL_C	1.0000	0.3654	0.6667	1.0000
all / CL_P+CL_C	0.6300	0.7196	0.8090	0.7992

the learning mechanism. Additional classifiers could consider the data types of two classes, could use a thesaurus for finding synonym class names, or could use other measures like KL-distance or mutual information (joint entropy). Furthermore, instead of averaging the classifier predictions, the weights of each classifier could be learned via regression. Another interesting direction of investigation would be to evaluate the effect to integrate our model with graph-matching algorithms like, for instance [23, 30]. These can be considered as additional classifiers. Last, but not least it would be interesting to evaluate the effect of allowing more expressive mapping rules, as for instance of the form $\alpha_{j,i} T_j(x) \leftarrow S_{i_1}(x), \ldots, S_{i_n}$(x) or more generally on full featured logic programming or of the form presented in [5], as well as to consider the impact of probabilities $Pr(\bar{S}_i|T_j)$, $Pr(\bar{S}_i|\bar{T}_j)$, $Pr(S_i|\bar{T}_j)$ [9] (and vice-versa inverting T_j with S_i).

[9] $\bar{X}$ is the complement of X.

Table 11. Traditional recall, All constraints

	f_{id}	f_{sum}	$f_{lin} \circ f_{sum}$	$f_{log} \circ f_{sum}$
CL_S	0.5785	0.5785	0.5785	0.5785
CL_{N-clm}	0.5430	0.4744	0.4652	0.4259
CL_{PN-clm}	0.3304	0.0100	0.0100	0.0100
CL_{kNN}	0.0200	0.0870	0.0285	0.0778
CL_B	0.0000	0.0763	0.0578	0.0093
all	0.5907	0.5815	0.4959	0.4959
CL_S / CL_P	0.5600	0.5600	0.5415	0.5415
CL_{N-clm} / CL_P	0.5622	0.4944	0.4181	0.3589
CL_{PN-clm} / CL_P	0.3296	0.0385	0.0193	0.0193
CL_{kNN} / CL_P	0.0385	0.0378	0.0478	0.1170
CL_B / CL_P	0.0193	0.0856	0.0293	0.0193
all / CL_P	0.6000	0.5707	0.4581	0.4381
CL_S / CL_C	0.5600	0.5600	0.5322	0.5322
CL_{N-clm} / CL_C	0.5530	0.4652	0.3611	0.3404
CL_{PN-clm} / CL_C	0.3696	0.0100	0.0100	0.0100
CL_{kNN} / CL_C	0.0100	0.0393	0.0100	0.0685
CL_B / CL_C	0.0100	0.0670	0.0300	0.0100
all / CL_C	0.5807	0.5622	0.4581	0.4381
CL_S / CL_P+CL_C	0.5515	0.5515	0.5237	0.5137
CL_{N-clm} / CL_P+CL_C	0.5722	0.4652	0.2948	0.2919
CL_{PN-clm} / CL_P+CL_C	0.3889	0.0293	0.0193	0.0293
CL_{kNN} / CL_P+CL_C	0.0193	0.0285	0.0193	0.0478
CL_B / CL_P+CL_C	0.0193	0.0670	0.0193	0.0193
all / CL_P+CL_C	0.6100	0.5344	0.4289	0.4189

References

1. Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.
2. Ricardo A. Baeza-Yates and Berthier Ribeiro-Neto. *Modern Information Retrieval*. Addison-Wesley Longman Publishing Co., Inc., 1999.
3. J. Berlin and A. Motro. Database schema matching using machine learning with feature selection. In *In Proceedings of the Conf. on Advanced Information Systems Engineering (CAiSE), 2002.*, 2002.
4. Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Moshe Y. Vardi. Lossless regular views. In *Proc. of the 21st ACM SIGACT SIGMOD SIGART Sym. on Principles of Database Systems (PODS 2002)*, pages 247–258, 2002.
5. Hans Chalupsky. Ontomorph: a translation system for symbolic knowledge. In *Proceedings of the 9th International Conference on Principles of Knowledge Representation and Reasoning (KR-04)*. AAAI Press, 2000.

Table 12. Hierarchical precision, All constraints

	f_{id}	f_{sum}	$f_{lin} \circ f_{sum}$	$f_{log} \circ f_{sum}$
CL_S	0.8053	0.8053	0.8053	0.7859
CL_{N-clm}	0.5800	0.5100	0.5087	0.5615
CL_{PN-clm}	0.3683	0.1409	0.5000	0.0714
CL_{kNN}	0.5000	0.6241	0.5833	0.2547
CL_B	0.0000	0.3562	0.3704	0.1667
all	0.6433	0.6958	0.7571	0.7514
CL_S / CL_P	0.7903	0.7946	0.8094	0.7890
CL_{N-clm} / CL_P	0.6000	0.5300	0.6934	0.7054
CL_{PN-clm} / CL_P	0.3733	0.2917	0.6250	0.7500
CL_{kNN} / CL_P	1.0000	0.6333	1.0000	0.3249
CL_B / CL_P	1.0000	0.3980	0.4167	0.7500
all / CL_P	0.6533	0.7241	0.8118	0.8071
CL_S / CL_C	0.8107	0.8150	0.8018	0.8018
CL_{N-clm} / CL_C	0.5950	0.5096	0.5972	0.6248
CL_{PN-clm} / CL_C	0.4133	0.3214	0.7500	0.3333
CL_{kNN} / CL_C	0.7500	0.7292	0.6250	0.2588
CL_B / CL_C	0.7500	0.4062	0.3250	0.6250
all / CL_C	0.6333	0.7237	0.8118	0.7956
CL_S / CL_P+CL_C	0.8181	0.8226	0.8376	0.8357
CL_{N-clm} / CL_P+CL_C	0.6100	0.6000	0.6952	0.7396
CL_{PN-clm} / CL_P+CL_C	0.4283	1.0000	1.0000	1.0000
CL_{kNN} / CL_P+CL_C	1.0000	0.8750	1.0000	0.7500
CL_B / CL_P+CL_C	1.0000	0.4351	0.6667	1.0000
all / CL_P+CL_C	0.6633	0.7417	0.8090	0.8117

6. Robin Dhamankar, Yoonkyong Lee, AnHai Doan, Alon Halevy, and Pedro Domingos. iMAP: discovering complex semantic matches between database schemas. In *Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, pages 383–394. ACM Press, 2004.
7. H. Do and E. Rahm. Coma - a system for flexible combination of schema matching approaches. In *Proceedings of the Int. Conf. on Very Large Data Bases (VLDB-02), 2002.*, 2002.
8. Anhai Doan, Pedro Domingos, and Alon Halevy. Learning to match the schemas of data sources: A multistrategy approach. *Mach. Learn.*, 50(3):279–301, 2003.
9. AnHai Doan, Pedro Domingos, and Alon Y. Halevy. Reconciling schemas of disparate data sources: a machine-learning approach. In *Proceedings of the 2001 ACM SIGMOD international conference on Management of data*, pages 509–520. ACM Press, 2001.
10. AnHai Doan, Jayant Madhavan, Robin Dhamankar, Pedro Domingos, and Alon Halevy. Learning to match ontologies on the semantic web. *The VLDB Journal*, 12(4):303–319, 2003.
11. AnHai Doan, Jayant Madhavan, Pedro Domingos, and Alon Halevy. Learning to map between ontologies on the semantic web. In *Proceedings of the eleventh*

Table 13. Hierarchical recall, All constraints

	f_{id}	f_{sum}	$f_{lin} \circ f_{sum}$	$f_{log} \circ f_{sum}$
CL_S	0.5970	0.5970	0.5970	0.5970
CL_{N-clm}	0.5615	0.4930	0.4791	0.4306
CL_{PN-clm}	0.3573	0.0177	0.0146	0.0100
CL_{kNN}	0.0200	0.1140	0.0509	0.1194
CL_B	0.0000	0.1043	0.0757	0.0093
all	0.6220	0.6083	0.5052	0.5006
CL_S / CL_P	0.5785	0.5739	0.5554	0.5554
CL_{N-clm} / CL_P	0.5807	0.5130	0.4181	0.3589
CL_{PN-clm} / CL_P	0.3616	0.0385	0.0193	0.0193
CL_{kNN} / CL_P	0.0385	0.0555	0.0478	0.1301
CL_B / CL_P	0.0193	0.1085	0.0426	0.0193
all / CL_P	0.6312	0.5930	0.4628	0.4428
CL_S / CL_C	0.5785	0.5739	0.5507	0.5507
CL_{N-clm} / CL_C	0.5761	0.4883	0.3657	0.3450
CL_{PN-clm} / CL_C	0.4012	0.0146	0.0146	0.0146
CL_{kNN} / CL_C	0.0146	0.0566	0.0196	0.1020
CL_B / CL_C	0.0146	0.0963	0.0446	0.0146
all / CL_C	0.6120	0.5811	0.4628	0.4428
CL_S / CL_P+CL_C	0.5700	0.5654	0.5376	0.5276
CL_{N-clm} / CL_P+CL_C	0.5907	0.4744	0.2948	0.2919
CL_{PN-clm} / CL_P+CL_C	0.4159	0.0293	0.0193	0.0293
CL_{kNN} / CL_P+CL_C	0.0193	0.0335	0.0193	0.0478
CL_B / CL_P+CL_C	0.0193	0.0817	0.0193	0.0193
all / CL_P+CL_C	0.6412	0.5487	0.4289	0.4235

Table 14. Overall traditional precision, recall and F-Measure

	Precision	Recall	F-Measure
No constraint	0.0919	0.5982	0.1307
Constraint 1	0.0939	0.5970	0.1323
Constraint 2	0.1495	0.4677	0.1856
Constraint 3	0.2185	0.4365	0.2621
Constraint 4	0.4995	0.3451	0.3012
Constraint 5	0.3821	0.3676	0.3745
All constraints	0.5979	0.2819	0.3284

international conference on World Wide Web, pages 662–673. ACM Press, 2002.
12. Thomas Eiter, Thomas Lukasiewicz, Roman Schindlauer, and Hans Tompits. Combining answer set programming with description logics for the semantic web. In *Proceedings of the 9th International Conference on Principles of Knowledge Representation and Reasoning (KR-04)*. AAAI Press, 2004.
13. David W. Embley, David Jackman, and Li Xu. Multifaceted exploitation of metadata for attribute match discovery in information integration. In *Workshop on Information Integration on the Web*, pages 110–117, 2001.
14. Ronald Fagin, Phokion G. Kolaitis, Reneé Miller, and Lucian Popa. Data exchange: Semantics and query answering. In *Proceedings of the International Conference on Database Theory (ICDT-03)*, number 2572 in Lecture Notes in Computer Science, pages 207–224. Springer Verlag, 2003.
15. Ronald Fagin, Phokion G. Kolaitis, and Lucian Popa. Data exchange: getting to the core. In *Proceedings of the twenty-second ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 90–101. ACM Press, 2003.
16. Ronald Fagin, Phokion G. Kolaitis, Wang-Chiew Tan, and Lucian Popa. Composing schema mappings: Second-order dependencies to the rescue. In *Proceedings PODS*, 2004.
17. Norbert Fuhr. Probabilistic Datalog: Implementing logical information retrieval for advanced applications. *Journal of the American Society for Information Science*, 51(2):95–110, 2000.
18. MingChuan Guo and Yong Yu. Mutual enhancement of schema mapping and data mapping. In *In ACM SIGKDD 2004 Workshop on Mining for and from the Semantic Web*, Seattle, 2004.
19. Bin He and Kevin Chen-Chuan Chang. Statistical schema matching across web query interfaces. In *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, pages 217–228. ACM Press, 2003.
20. Ian Horrocks and Peter F. Patel-Schneider. A proposal for an OWL rules language. In *Proc. of the Thirteenth International World Wide Web Conference (WWW-04)*. ACM, 2004.
21. Ian Horrocks, Peter F. Patel-Schneider, and Frank van Harmelen. From SHIQ and RDF to OWL: The making of a web ontology language. *Journal of Web Semantics*, 1(1):7–26, 2003.
22. R. Ichise, H. Takeda, and S. Honiden. Rule induction for concept hierarchy alignment. In *Proceedings of the Workshop on Ontology Learning at the 17th International Joint Conference on Artificial Intelligence (IJCAI-01)*, 2001.
23. Jaewoo Kang and Jeffrey F. Naughton. On schema matching with opaque column names and data values. In *Proceedings of the 2003 ACM SIGMOD international conference on on Management of data*, pages 205–216. ACM Press, 2003.
24. Martin S. Lacher and Georg Groh. Facilitating the exchange of explicit knowledge through ontology mappings. In *Proceedings of the Fourteenth International Florida Artificial Intelligence Research Society Conference*, pages 305–309. AAAI Press, 2001.
25. Maurizio Lenzerini. Data integration: a theoretical perspective. In *Proceedings of the 21st ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems (PODS-02)*, pages 233–246. ACM Press, 2002.
26. John W. Lloyd. *Foundations of Logic Programming*. Springer, Heidelberg, RG, 1987.

27. J. Madhavan, P. Bernstein, K. Chen, A. Halevy, and P. Shenoy. Corpus-based schema matching. In *Workshop on Information Integration on the Web at IJCAI-03*, 2003.
28. Jayant Madhavan, Philip A. Bernstein, and Erhard Rahm. Generic schema matching with cupid. In *Proc. 27th VLDB Conference*, pages 49–58, 2001.
29. Ronald Fagin Marcelo Arenas, Pablo Barcelo and Leonid Libkin. Locally consistent transformations and query answering in data exchange. In *Proceedings of the 23st ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems (PODS-04)*, pages 229–240. ACM Press, 2004.
30. S. Melnik, H. Garcia-Molina, and E. Rahm. Similarity flooding: A versatile graph matching algorithm and its application to schema matching. In *Proceedings of the 18th International Conference on Data Engineering (ICDE'02)*, page 117. IEEE Computer Society, 2002.
31. Henrik Nottelmann and Umberto Straccia. A probabilistic approach to schema matching. Technical Report 2004-TR-60, Istituto di Scienza e Tecnologie dell'Informazione, Consiglio Nazionale delle Ricerche, Pisa, Italy, 2004.
32. Natalya Fridman Noy and Mark A. Musen. Prompt: Algorithm and tool for automated ontology merging and alignment. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*, pages 450–455. AAAI Press / The MIT Press, 2000.
33. Lucian Popa, Yannis Velegrakis, Renee J. Miller, Mauricio A. Hernandez, and Ronald Fagin. Translating web data. In *Proceedings of VLDB 2002, Hong Kong SAR, China*, pages 598–609, 2002.
34. Erhard Rahm and Philip A. Bernstein. A survey of approaches to automatic schema matching. *The VLDB Journal*, 10(4):334–350, 2001.
35. Fabrizio Sebastiani. Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1):1–47, 2002.
36. Ling Ling Yan, Renée J. Miller, Laura M. Haas, and Ronald Fagin. Data-driven understanding and refinement of schema mappings. In *Proceedings of the 2001 ACM SIGMOD international conference on Management of data*, pages 485–496. ACM Press, 2001.