Available online at www.sciencedirect.com

ScienceDirect

Information Processing and Management 43 (2007) 552–576

INFORMATION PROCESSING & MANAGEMENT

www.elsevier.com/locate/infoproman

Information retrieval and machine learning for probabilistic schema matching

Henrik Nottelmann [a,✠], Umberto Straccia [b,*]

[a] Department of Informatics, University of Duisburg-Essen, 47048 Duisburg, Germany
[b] ISTI-CNR, Via G. Moruzzi 1, 56124 Pisa, Italy

Received 19 December 2005; received in revised form 23 October 2006; accepted 23 October 2006
Available online 8 December 2006

"I deeply regret the sudden death of Henrik Nottelmann, a nice colleague, talented young scientist and good friend"

Abstract

Schema matching is the problem of finding correspondences (mapping rules, e.g. logical formulae) between heterogeneous schemas e.g. in the data exchange domain, or for distributed IR in federated digital libraries. This paper introduces a probabilistic framework, called sPLMap, for automatically learning schema mapping rules, based on given instances of both schemas. Different techniques, mostly from the IR and machine learning fields, are combined for finding suitable mapping candidates. Our approach gives a probabilistic interpretation of the prediction weights of the candidates, selects the rule set with highest matching probability, and outputs probabilistic rules which are capable to deal with the intrinsic uncertainty of the mapping process. Our approach with different variants has been evaluated on several test sets.
© 2006 Elsevier Ltd. All rights reserved.

Keywords: Schema matching; Data exchange; Probability theory; sPLMap

1. Introduction

Distributed information systems tend to be highly heterogeneous, integrate different computer platforms, data storage formats, document models and schemas which structure the documents. The latter aspect requires to transform data structured under one schema into data structured under a different schema. This old but emerging problem is particularly interesting from an information retrieval perspective for two reasons: First, the intrinsic uncertainty of finding mappings between different schemas has to be taken into account. Second, typically techniques from the information retrieval and machine learning areas are employed. The problem of automatically finding mappings between schemas is called *schema matching* (Rahm & Bernstein,

* Corresponding author.
E-mail address: straccia@isti.cnr.it (U. Straccia).
✠ Deceased.

0306-4573/$ - see front matter © 2006 Elsevier Ltd. All rights reserved.
doi:10.1016/j.ipm.2006.10.014

2001), and is currently under investigation in the context of *information integration* (Lenzerini, 2002) and *data exchange* (Fagin, Kolaitis, Miller, & Popa, 2003) (see Section 5 for a detailed description).

One important application is information retrieval in federated digital libraries or peer-to-peer networks (see MIND and PEPPER projects[1]). Heterogeneity of schemas is one of the key problems in these environments, thus automatic schema matching is a major issue. Often it is infeasible to materialize the sources in the global schema, e.g. because the libraries are maintained by a third party like the ACM DL or CiteSeer, or because it is too expensive to acquire and convert all documents in advance. Thus, a retrieval run typically involves five steps:

(1) selecting relevant information sources;
(2) rewriting the user query w.r.t. the local schemas, using the mapping rules;
(3) retrieving documents from the local sources using the rewritten query;
(4) transforming documents into the global schema (using the same mapping rules in the other direction); and
(5) filtering documents in cases where the query transformation is incomplete, and thus non-matching documents are returned.

The schema matching framework sPLMap (probabilistic, logic-based mapping between schemas) presented in this paper combines ideas from data integration and data exchange, from information retrieval and machine learning. The global (target) schema (which can be presented to a user) is fixed and defined independently from other schemas. *Mapping rules*, such as (informally)

$$0.98\ \texttt{standard\_booktitle} \leftarrow \texttt{BIBDB\_journal} \tag{1}$$

describe correspondences between attributes of these schemas, and are specified using high-level declarative (e.g. logical) formalisms. For instance, in the above example, the target schema is called `standard`, while the source schema is `BIBDB`. The mapping above expresses that the `journal` attribute of the `BIBDB` schema can be mapped into the `booktitle` attribute of the `standard` schema with probability 0.98.

From well-known approaches like LSD (Doan, Domingos, & Halevy, 2001), sPLMap borrows the idea of combining several specialized components (called "classifiers") for finding the best mapping, e.g. based on attribute names or comparing properties of the underlying data instances. The major improvements over LSD are the support for data types (e.g. text, names, different date formats) in the matching process, the usage of probability theory for estimating the degree of correctness of a set of learned rules, and the computation of probabilistic rules for capturing the inherent uncertainty of the matching process. Thus, it provides a sound theoretical justification for its (optimum) selection, and a measure for the quality of a learned schema mapping.

A preliminary variant of sPLMap has been described in Nottelmann and Straccia (2005), together with some first evaluation results. This paper contains several extensions: We show how the learned mapping rules can be employed for query transformation; we introduce additional classifiers, we investigate the effect of learning the weights for classifiers individually, and we present a larger evaluation on four different test beds.

Worth mentioning is also that the same principles of this work have been used in ontology and Web directory alignment (Nottelmann & Straccia, 2006; Straccia & Troncy, 2005), where the objective was to find correspondences between the categories of the source ontology/Web directory and the target ontology/Web directory. For instance, an excerpt of two "university course" ontologies is given in Fig. 1. It also reports the mappings between the categories of the two ontologies, which have been found automatically.

The paper is structured as follows: The next section introduces a formal framework for schema mapping. Section 3 presents a theoretically founded approach for learning these schema mappings, where the predictions of different classifiers are combined. This approach is evaluated on a large test bed in Section 4. Section 5 discusses related work in the context of sPLMap. The last section summarizes this paper and gives an outlook over future works.

[1] `http://www.mind-project.org`, `http://www.pepper-project.org/`.

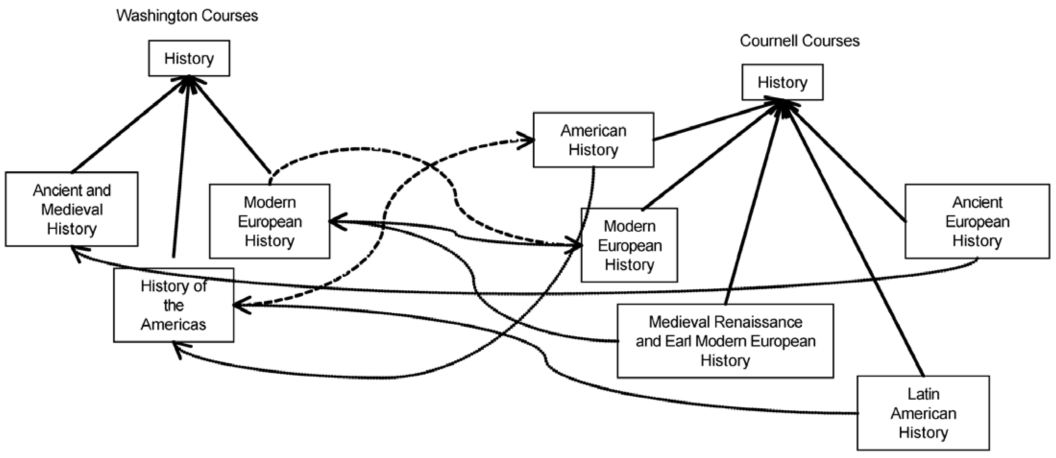

Fig. 1. The excerpt of two ontologies and category matchings.

2. Schema mapping

This section introduces a formal, logic-based framework for schema mapping, where the mapping process is fully automatic. It is inspired by the data exchange problem (Fagin et al., 2003) and extends it to a framework capable of coping with the intrinsic uncertainty of the mapping process. sPLMap employs probabilistic Datalog (introduced in Section 2.1) for specifying mapping rules. These are of the form (see Section 2.3 for details):

$$\alpha_{j,i}T_j(d,v) \leftarrow S_i(d,v).$$

Informally, the above rule dictates that the attribute S_i of the source schema is mapped into the attribute T_j of the target schema with probability $\alpha_{j,i}$. More specifically, it says that for all documents d and value x of attribute S_i, x is also the value of attribute T_j with probability $\alpha_{j,i}$. For instance, assume that the source schema `BIBDB` has attributes

```
<document id>, <doctype>, <author>, <title>, <journal>
<volume>, <number>, <pages>, <year>
```

and the target schema `standard` has attributes

```
<document id>, <author>, <year>, <type>, <title>
<booktitle>, <misc>, <pages>
```

then a rule such as (the formal counterpart of the mapping (1))

$$0.98\ \texttt{standard\_booktitle(d, v)} \leftarrow \texttt{BIBDB\_journal(d, v)} \tag{2}$$

encodes the fact that likely the journal attribute in the source schema corresponds to the book title attribute in the target schema. Hence, a query about specific book titles in the `standard` schema is then translated into a query about journals in the `BIBDB` schema. We refer the reader to Fuhr (2000) for a detailed description on how query answering is performed in probabilistic Datalog. The focus in this paper is on how to learn these rules automatically.

2.1. Probabilistic datalog

In the following, we briefly describe Probabilistic Datalog (pDatalog for short) (Fuhr, 2000). 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, in order to simplify the presentation we will not deal with negation in this paper.

In pDatalog every fact or rule has a probabilistic weight $0 < \alpha \leqslant 1$ 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 $B_1, \ldots, B_n$ ($n \geqslant 0$) 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 any instantiation of rule r is true is α". The following example pDatalog program expresses the fact that in 80% of all cases, a year stored in document metadata is the publication year of that document:

$$0.8\ \texttt{pubyear}(d,y) \leftarrow \texttt{year}(d,y).$$

The facts

$$\begin{aligned}&\texttt{year(d1, 2002)},\\&0.5\ \texttt{year(d2, 2003)}\end{aligned}$$

(the weight might be derived e.g. during an automatic information extraction process, which is inherently uncertain) imply

$$\begin{aligned}&Pr(\texttt{pubyear(d1, 2002)}) = 0.8,\\&Pr(\texttt{pubyear(d2, 2003)}) = 0.8 \cdot 0.5 = 0.4.\end{aligned}$$

Another example is where we have the rules:

$$\begin{aligned}&\texttt{author}(d,n) \leftarrow \texttt{contact\_author}(d,n),\\&\texttt{author}(d,n) \leftarrow \texttt{co\_author}(d,n)\end{aligned}$$

dictating that an author of a document is either the contact author or a co-author. Hence, if we have the facts stating that we are unsure about whether N is the contact author (with probability 70%) or the co-author (with probability 60%) of document d1, i.e. the facts

$$\begin{aligned}&0.7\ \texttt{author(d1, N)},\\&0.6\ \texttt{co\_author(d1, N)},\end{aligned}$$

then we may infer that N is an author with probability $0.88 = 0.7 + 0.6 - 0.7 \cdot 0.6$, under the independence assumption (as there are two alternatives to infer the same information).

For the ease of presentation, in the following we consider a succinct representation of the probabilistic program above. It may help the reader to follow the definition of the semantics of probabilistic Datalog. So, let us consider the rules (representing "a if either b or c", and a,b,c stand for author, contact_author and co_author, respectively)

$$\begin{aligned}&\texttt{a} \leftarrow \texttt{b},\\&\texttt{a} \leftarrow \texttt{c}\end{aligned}$$

and the probabilistic facts ("b is true with probability 70%, while c is true with probability 60%")

$$\begin{aligned}&0.7\texttt{b},\\&0.6\texttt{c}.\end{aligned}$$

Formally, an *interpretation structure* in pDatalog is a tuple $\mathscr{I} = (\mathscr{W}, \mu)$, where $\mathscr{W}$ is a set of possible worlds and μ is a probability distribution over $\mathscr{W}$. The possible worlds are defined as follows. Given a pDatalog program P, with $H(P)$ we indicate the ground instantiation of P.[2] In our case, we have $H(P) = P$, as P does

[2] 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*.

not have any variables. Then, the *deterministic part* of P is the set P_D of instantiated rules in $H(P)$ having weight $\alpha = 1$. In our example, P_D contains two rules

$$P_D = \{(\mathtt{a} \leftarrow \mathtt{b}), (\mathtt{a} \leftarrow \mathtt{c})\}.$$

The *indeterministic part* of P is the set P_I of instantiated rules determined by $P_I = \{r : \alpha r \in H(P),\ \alpha < 1\}$. In our example,

$$P_I = \{\mathtt{b}, \mathtt{c}\}.$$

The set of deterministic programs of P, denoted $D(P)$ is defined as $D(P) = \{P_D \cup Y : Y \subseteq P_I\}$. In our example,

$$D(P) = \{P_1, P_2, P_3, P_4\},$$

where the programs $P_1, \ldots, P_4$ are

$$\begin{aligned} P_1 &= P_D \cup \emptyset = P_D, \\ P_2 &= P_D \cup \{\mathtt{b}\}, \\ P_3 &= P_D \cup \{\mathtt{c}\}, \\ P_4 &= P_D \cup \{\mathtt{b}, \mathtt{c}\}. \end{aligned}$$

Note that any $P' \in D(P)$ is a classical logic program. Finally, a possible world $w \in \mathscr{W}$ is the minimal model (Lloyd, 1987) 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*). In our example, for each of the four programs $P_i \in D(P)$ we have a minimal model w_i: the models $w_1, \ldots, w_4$ are

$$\begin{aligned} w_1 &= \emptyset, \\ w_2 &= \{\mathtt{a}, \mathtt{b}\}, \\ w_3 &= \{\mathtt{a}, \mathtt{c}\}, \\ w_4 &= \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}. \end{aligned}$$

w_1 is a model of P_1 as no fact is known in P_1. w_2 is a model of P_2 as it satisfies the fact $\mathtt{b} \in P_1$ and, thus, $\mathtt{a}$ has to hold as well (due to the rule $\mathtt{a} \leftarrow \mathtt{b}$). The cases of w_3 and w_4 are similar. Hence, the set of worlds for P is $\mathscr{W} = \{w_1, w_2, w_3, w_4\}$.

Now, an *interpretation* is a tuple $I = (\mathscr{I}, w)$ such that $w \in \mathscr{W}$. The notion of truth w.r.t. an interpretation and a possible world can be defined recursively:

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

In our example, consider the interpretation $I = (\mathscr{I}, w_2)$, with $\mathscr{I} = (\mathscr{W}, \mu)$, $\mathscr{W} = \{w_1, w_2, w_3, w_4\}$ and the distribution μ

$$\begin{aligned} \mu(w_1) &= 0.08, \\ \mu(w_2) &= 0.32, \\ \mu(w_3) &= 0.22, \\ \mu(w_4) &= 0.38. \end{aligned}$$

Then, it can be verified that

$$\begin{aligned}
&(\mathcal{I}, w_2) \models \mathtt{b},\\
&(\mathcal{I}, w_2) \models \mathtt{a},\\
&(\mathcal{I}, w_2) \models \mathtt{a} \leftarrow \mathtt{b},\\
&(\mathcal{I}, w_2) \models \mathtt{a} \leftarrow \mathtt{c},\\
&(\mathcal{I}, w_2) \models 0.7\mathtt{b},\\
&(\mathcal{I}, w_2) \models 0.6\mathtt{c},\\
&(\mathcal{I}, w_2) \models 0.92\mathtt{a}.
\end{aligned}$$

Note that $(\mathcal{I}, w_2) \models 0.7\mathtt{b}$, as the set of worlds satisfying $\mathtt{b}$ is $\{w_2, w_4\}$ and $\mu(\{w_2, w_4\}) = 0.32 + 0.38 = 0.7$. Similarly, $(\mathcal{I}, w_2) \models 0.92\mathtt{a}$, as the set of worlds satisfying $\mathtt{a}$ is $\{w_2, w_3, w_4\}$ and $\mu(\{w_2, w_3, w_4\}) = 0.32 + 0.22 + 0.38 = 0.92$.

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

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

In our example, the interpretation just defined above is a model of P. Below, we report a list of models I_i of our example program P. We report just the probability distribution μ. The bottom row illustrates the probability α that the fact $\mathtt{a}$ is true, i.e $\alpha = \mu(w_2) + \mu(w_3) + \mu(w_4)$.

$\mu(w_i)$	I_1	I_2	I_3	…
$\mu(w_1)$	0.0	0.12	0.08	…
$\mu(w_2)$	0.4	0.28	0.32	…
$\mu(w_3)$	0.3	0.18	0.22	…
$\mu(w_4)$	0.3	0.42	0.38	…
α	1.0	0.88	0.92	…

It can be verified that for any model I_i of P, we have that $0.88 \leqslant \alpha \leqslant 1.0$. That is, in any model of P, the atom $\mathtt{a}$ is true at least with probability 0.88.

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, A is true with probability at least α. For instance, in our example, we have that

$$\begin{aligned}
&P \models 0.7\ \mathtt{b},\\
&P \models 0.6\ \mathtt{c},\\
&P \models 0.88\ \mathtt{a}.
\end{aligned}$$

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.

In the remainder, given an n-ary atom A for predicate $\overline{A}$ and an interpretation $I = (\mathcal{I}, w)$, with A^I (an *instantiation* of A w.r.t. the interpretation I) we indicate the set of ground facts $\alpha\overline{A}(c_1, \ldots, c_n)$, where the ground atom $\overline{A}(c_1, \ldots, c_n)$ is contained in the world w, and $\mu(\{w' \in \mathcal{W} : (\mathcal{I}, w') \models \overline{A}(c_1, \ldots, c_n)\}) = \alpha$, i.e. $I \models \alpha\overline{A}(c_1, \ldots, c_n)$. Essentially, A^I is the set of all instantiations of A under I with relative probabilities, i.e. under I, $\overline{A}(c_1, \ldots, c_n)$ holds with probability α.

2.2. Data types and schemas

We first assume a finite set $\mathbf{D}$ of elementary *data types*. The domain *dom*(d) for a data type $d \in \mathbf{D}$ defines the set of possible values for d. Examples for data types are `Text` (for English text), `Name` (person names, e.g. "John Doe"), `Year` (four digit year numbers, e.g. "2004"), `DateISO8601` for the ISO 8601 format of dates (e.g. "2004-12-31") or `DateEnglish` (e.g. "Dec 12, 2004"). A fixed data type `DOCID` $\in \mathbf{D}$ is used for document identifiers.

For sPLMap, we consider a simple yet effective structure for schemas as a linear list of multi-valued schema attributes. Each attribute can be modelled as a binary relation, which stores pairs of a document id and a value for that attribute. It is not our purpose here to address the case where schemas are defined using more expressive languages like XML. Examples of our schema model are BibTeX or Dublin Core.[3]

Formally, a *schema* $\mathbf{R} = \langle R_1, \ldots, R_n \rangle$ consists of a non-empty finite tuple of binary relation symbols (corresponding to *attributes* of a document). Each relation symbol R_i has a data type $d_{R_i} \in \mathbf{D}$. A schema (or a single schema attribute) can be instantiated with tuples forming the content of documents. An *instance* of schema attribute R_j is then a set of probabilistic ground facts:

$$R_j^I \subseteq \{\alpha R_j(d, v) : \alpha \in [0,1], d \in \mathtt{DOCID}, v \in dom(d_{R_i})\}.$$

For instance, the `BIBDB` schema is made out by the attributes `documentid`, `author`, `year`, `type`, `title`, `booktitle`, `misc` and `pages`, each of which is of type `String`. An actual document is of the form

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<document id="BIBDB.r0/1">
<doctype>article</doctype>
<author>Imielinski, T.</author>
<title>Incomplete deductive databases</title>
<journal>Annals of Mathematics and Artificial Intelligence</journal>
<volume>3</volume>
<number>2-4</number>
<pages>259–294</pages>
<year>1991</year>
</document>
```

and will be encoded as the pDatalog program made out by the following facts ($\alpha = 1$, always)

```
doctype(BIBDB.r0/1,"article")
author(BIBDB.r0/1,"Imielinski, T.")
title(BIBDB.r0/1,"Incomplete deductive databases")
journal(BIBDB.r0/1,"Annals o Mathematics and Artificial Intelligence")
volume(BIBDB.r0/1,"3")
number(BIBDB.r0/1,"2-4")
pages(BIBDB.r0/1,"259-294")
year(BIBDB.r0/1,"1991")
```

In our approach, an instance of the source schema is obtained directly from a set of source documents. Therefore, e.g. `author(BIBDB.r0/1,"Imielinski, T.")` may belong to the instance of the attribute `author`, $\mathtt{author}^I$.

For simplicity, we use R_j for both the schema attribute R_j and its instances R_j^I, and similarly $\mathbf{R}$ for the schema and the corresponding *schema instance*.

The following example schema will be used in the remainder of this section for illustrative purpose:

$$\mathbf{A} = \langle \mathtt{creator}, \quad \mathtt{date} \rangle, d_{\mathtt{creator}} = \mathtt{Name}, \quad d_{\mathtt{date}} = \mathtt{Date},$$
$$\mathbf{B} = \langle \mathtt{author}, \mathtt{editor}, \mathtt{created} \rangle, \quad d_{\mathtt{author}} = d_{\mathtt{editor}} = \mathtt{Name}, \quad d_{\mathtt{created}} = \mathtt{Date}.$$

In this case, schema **A** has two attributes, `creator` storing the creator of a document, and `date` storing the publication date. On the other hand, schema **B** has a finer granularity, as it keeps the author and the editor in separate attributes. Obviously, the attribute `created` corresponds to attribute `date` in schema **A**.

[3] http://dublincore.org/.

2.3. Schema mappings

Let us assume a source schema $\mathbf{S} = \langle S_1, \ldots, S_s \rangle$, a target schema $\mathbf{T} = \langle T_1, \ldots, T_t \rangle$, where **S** and **T** are disjoint, and a set Σ of mapping rules that specify how and what source data should appear in the target schema. In this paper, probabilistic Datalog rules of the form

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

are employed, such as (2), denoting that content from a source schema attribute S_i has to be copied to a target attribute T_j with a probability of $\alpha_{j,i}$.

The schema matching problem is then stated as follows: Given instances of source schema **S** and target schema **T**, find an appropriate set Σ of mapping rules. More precisely, find the Σ which maximizes the probability that a target instance **T** and the result of applying the mapping rules Σ onto the source instance **S** are "similar". We will see later on how we compute the similarity. The tuple $\mathcal{M} = (\mathbf{T}, \mathbf{S}, \Sigma)$ is called a *mapping*. The ultimate goal of sPLMap is to automatically determine the relationships between attributes of the two schemas.

For a set Σ_j of mapping rules with common relation T_j in the rule heads, the result of applying these rules onto the source instance **S** yields an instance over the target attribute T_j, the *mapping result*. This instance is denoted by $\widehat{T_j}$. $\widehat{\mathbf{T}} = \langle \widehat{T_1}, \ldots, \widehat{T_t} \rangle$ denotes the schema instance derived by applying Σ onto **T**. Typically, the set Σ of mapping rules on which $\widehat{T_j}$ and $\widehat{\mathbf{T}}$ depend is clear from the context and left out; in some exceptional cases, we explicitly mention it as $\widehat{T_j}^{\Sigma}$ and $\widehat{\mathbf{T}}^{\Sigma}$, respectively.

Note that a source attribute may be mapped into several target attributes and a target attribute may be the target of many source attributes, i.e. we may have complex mappings. For example,

$$\Sigma = \{\alpha_{1,1} T_1(d, v) \leftarrow S_1(d, v), \alpha_{1,2} T_1(d, v) \leftarrow S_2(d, v), \alpha_{2,1} T_2(d, v) \leftarrow S_1(d, v)\}$$

is a legal set of mapping rules. However, sPLMap does not require that a mapping exists for every target attribute.

Reasonable rules for the example schemas $\mathbf{T} = \mathbf{A}$ and $\mathbf{S} = \mathbf{B}$, where the attribute `creator` subsumes both author and editor, are:

$$\begin{aligned}
&\mathtt{creator}(d, v) \leftarrow \mathtt{author}(d, v),\\
&\mathtt{creator}(d, v) \leftarrow \mathtt{editor}(d, v),\\
&\mathtt{date}(d, v) \leftarrow \mathtt{created}(d, v).
\end{aligned}$$

Now, we restrict the example to the mapping $\Sigma = \{\mathtt{creator} \leftarrow \mathtt{author}\}$. Then, the following source instances `author` and `editor` define the resulting target instance with:

$$\begin{aligned}
&\mathtt{author} = \{\mathtt{author(d1,"John\ Doe")}\},\\
&\mathtt{editor} = \{\mathtt{editor(d2,"Jane\ Meyer")}\},\\
&\widehat{\mathtt{creator}} = \{\mathtt{creator(d1,"John\ Doe")}, \mathtt{creator(d2,"Jane\ Meyer")}\}.
\end{aligned}$$

For the other direction, i.e. $\mathbf{T} = \mathbf{B}$ and $\mathbf{S} = \mathbf{A}$, possible rules are:

$$\begin{aligned}
&0.7\ \mathtt{author}(d, v) \leftarrow \mathtt{creator}(d, v),\\
&0.3\ \mathtt{editor}(d, v) \leftarrow \mathtt{creator}(d, v),\\
&\mathtt{created}(d, v) \leftarrow \mathtt{date}(d, v).
\end{aligned}$$

In other words, creators are authors with a probability of 70%, and editors with a probability of 30%.

Possible instances are then:

$$\begin{aligned}
&\mathtt{creator} = \{\mathtt{creator(d1,"John\ Doe")}\},\\
&\widehat{\mathtt{author}} = \{0.7\ \mathtt{author(d1,"John\ Doe")}\},\\
&\widehat{\mathtt{editor}} = \{0.3\ \mathtt{editor(d1,"John\ Doe")}\},
\end{aligned}$$

2.4. Queries and query transformation

A query q is a set Q of pDatalog rules with common head which define a unary predicate q with $q^I \subseteq \{\alpha q(d) : \alpha \in [0,1], d \in \mathtt{DOCID}\}$. The literals of these rules refer to the relation symbols defined in $\mathbf{R}$. The set $\bar{q}(\mathbf{R})$ of answers for query q with respect to $\mathbf{R}$ contains exactly all the document ids which satisfy the query:

$$\bar{q}(\mathbf{R}) := \{\alpha\bar{q}(d) | \mathbf{R}^I \cup Q \models \alpha\bar{q}(d)\}.$$

Given a schema mapping $\mathscr{M}$ and a source schema instance $\mathbf{S}$, the set $\bar{q}(\mathscr{M}, \mathbf{S})$ of certain answers to a query q (over $\mathbf{T}$) with respect to $\mathscr{M}$ and $\mathbf{S}$ is exactly the set of answers for that query w.r.t. the mapping result $\widehat{\mathbf{T}}$:

$$\bar{q}(\mathscr{M}, \mathbf{S}) := \bar{q}(\widehat{\mathbf{T}}).$$

A query q' w.r.t. the source schema $\mathbf{S}$ is a perfect reformulation of a query q w.r.t. the target schema $\mathbf{T}$ if the answers are the same (modulo different relations):

$$\alpha q'(d) \in \bar{q'}(\mathbf{S}) \iff \alpha q(d) \in \bar{q}(\widehat{\mathbf{T}})$$

for a given source instance $\mathbf{S}$, a set Σ of mapping rules, and the mapping result $\widehat{\mathbf{T}}$. However, in this paper, we are interested in correct (i.e. sound) reformulations:

$$\alpha q'(d) \in \bar{q'}(\mathbf{S}) \Rightarrow \alpha q(d) \in \bar{q}(\widehat{\mathbf{T}}).$$

The facts in $\bar{q}(\mathbf{S})$ then are certain answers, but q' does not necessary return all certain answers. This subset-property allows for handling cases in which no exact query transformation is possible.

A given schema mapping $\mathscr{M} = (\mathbf{T}, \mathbf{S}, \Sigma)$ can also be used for transforming a query for the target schema $\mathbf{T}$ into a query for $\mathbf{S}$. For this, the query has to be unfolded, i.e. all sub goals in the query rule bodies which refer to a relation in $\mathbf{T}$ have to be replaced by the bodies of the mapping rules.

The following example illustrates the definitions above. Consider the situation $\mathbf{T} = \mathbf{A}$ and $\mathbf{S} = \mathbf{B}$, the mapping rules Σ from above, and the query

$$\mathtt{q}(d) \leftarrow \mathtt{creator}(d, \text{"}\mathtt{John\ Doe}\text{"}).$$

Essentially, we are looking for documents created by John Doe. Using the mapping rules, the query over the target schema $\mathbf{T}$ can be reformulated into two queries over the source schema $\mathbf{S}$:

$$\mathtt{q1}(d) \leftarrow \mathtt{author}(d, \text{"}\mathtt{John\ Doe}\text{"}),$$
$$\mathtt{q1}(d) \leftarrow \mathtt{editor}(d, \text{"}\mathtt{John\ Doe}\text{"}).$$

Here, answers to the rewritten query `q1` are identifiers of documents whose author or editor is John Doe. Note that alternative queries may be:

$$\mathtt{q2}(d) \leftarrow \mathtt{author}(d, \text{"}\mathtt{John\ Doe}\text{"}),$$
$$\mathtt{q3}(d) \leftarrow \mathtt{editor}(d, \text{"}\mathtt{John\ Doe}\text{"}),$$
$$\mathtt{q4}(d) \leftarrow .$$

With the instances from above, we have $\bar{q}(\widehat{\mathbf{T}}) = \{\mathtt{q(d1)}\}$. Then, `q1` and `q2` are perfect reformulations, `q3` is a correct reformulation (as $\bar{\mathtt{q3}}(\mathbf{S}) = \emptyset$), while `q4` is not a correct reformulation:

$$q4(d2) \in \bar{\mathtt{q4}}(\mathbf{S}), \quad q(d2) \notin \bar{\mathtt{q}}(\widehat{\mathbf{T}}).$$

For the other direction, the query

$$\mathtt{q}(d) \leftarrow \mathtt{author}(d, v)$$

can be unfolded into the perfect reformulation

$$0.7\mathtt{q}'(d) \leftarrow \mathtt{creator}(d, v).$$

3. Learning schema mappings

This paper only deals with learning schema mappings, i.e. finding associations between attributes. We assume that a set of data types **D** is already given. Learning schema mapping consists of four steps:

(1) For each single possible mapping $T_j(d,v) \leftarrow S_i(d,v)$, we estimate the quality, arriving at a probability $Pr(S_i|T_j)$ (the order of this conditional probability is explained below).
(2) We form each possible set Σ, and estimate its quality based on the quality measures for its constituent mapping rules.
(3) Among all possible sets Σ, we select the "best" schema mapping according to our quality measure.
(4) Finally, we compute probabilities $\alpha_{j,i} = Pr(T_j|S_i)$ for each mapping rule in the best Σ.

We are going to illustrate these steps in detail.

3.1. Estimating the quality of a mapping

Consider a target schema $\mathbf{T} = \langle T_1, \ldots, T_t\rangle$ and a source schema $\mathbf{S} = \langle S_1, \ldots, S_s\rangle$ together with two instances. As above, for simplicity we use T_j and S_i for both the schema attribute and its instances, respectively, and **T** and **S**, for the schemas and their instances, respectively.

The goal of schema matching is to find the best set Σ of mapping rules. In sPLMap, we assess the quality of mapping for deterministic rules only. The rules from the best mapping are augmented with probabilistic weights (for the document and query transformation processes) in a later step.

Thus, all rules in any set Σ of mappings are deterministic in this section. For simplicity, we assume deterministic target and source instances; in other words, all ground facts considered here are deterministic.[4] Then, an interpretation R_i^I can alternatively be described as a set of tuples which correspond to the instance's ground facts:

$$R_i^I \equiv \{t : R_i(t) \in R_i^I\}.$$

Essentially, we may see R_i^I as the set of all values t of attribute R_i. This setting will ease the description of the quality estimation step.

In the remainder of this paper, $\Sigma_{j,i}$ denotes the set containing the single (deterministic) rule $T_j(d,v) \leftarrow S_i(d,v)$. The $\Sigma_{j,i}$ form the building blocks of larger sets of mapping rules: A set $\Sigma_j \subseteq \cup_i \Sigma_{j,i}$ contains rules with common target attribute T_j, while Σ denotes a set of mapping rules with different target attributes:

$$\Sigma_{j,i} := \{T_j(d,v) \leftarrow S_i(d,v)\},$$
$$\Sigma_j = \bigcup_{i_k} \Sigma_{j,i_k},$$
$$\Sigma = \bigcup_j \Sigma_j.$$

When all instances are deterministic as in this section, the mapping result can easily be computed as

$$\widehat{T_j} = \widehat{T_j}^{\Sigma_j} = \bigcup_{i_k} \widehat{T_j}^{\Sigma_{j,i}} = \bigcup_{i_k} S_{i_k}. \tag{3}$$

In our example setting with

$T_1 = \texttt{creator}$,
$T_2 = \texttt{date}$,
$S_1 = \texttt{author}$,
$S_2 = \texttt{editor}$,
$S_3 = \texttt{created}$,

[4] Extension to uncertain instances is straightforward, but requires a more complex formalism to be described.

we have to consider single sets

$$\Sigma_{j,i}, \quad \text{with } j = 1, 2 \text{ and } i = 1, 2, 3,$$

e.g.

$$\Sigma_{1,2} = \{\texttt{creator}(d, v) \leftarrow \texttt{editor}(d, v)\}.$$

Then for all j, we have to consider all subsets Σ_j of $\Sigma_{j,1} \cup \Sigma_{j,2} \cup \Sigma_{j,3}$. For instance, we may consider the candidate rule sets:

$$\Sigma_1 = \{\texttt{creator}(d, v) \leftarrow \texttt{editor}(d, v)\},$$
$$\Sigma_2 = \{\texttt{date}(d, v) \leftarrow \texttt{editor}(d, v),\ \texttt{date}(d, v) \leftarrow \texttt{created}(d, v)\}.$$

(where Σ_2 is obviously wrong).

The goal is to find the "best" set of mapping rules Σ which maximizes the probability $Pr(\Sigma, \mathbf{T}, \mathbf{S})$ that the tuples in $\mathbf{T}$ and in the mapping result $\widehat{\mathbf{T}}$ are 'similar', that is, given a random tuple in $\mathbf{T}$, to estimate the probability that it is a tuple in $\widehat{\mathbf{T}}$ and vice-versa.

As mentioned above, Σ can be partitioned into t sets $\Sigma_1, \ldots, \Sigma_t$, and the rules in these sets operate independently on different target attributes:

$$Pr(\Sigma, \mathbf{T}, \mathbf{S}) = \prod_{j=1}^{t} Pr(\Sigma_j, \mathbf{T}, \mathbf{S}). \tag{4}$$

Now, $Pr(\Sigma_j, \mathbf{T}, \mathbf{S})$ is estimated as the probability that a tuple in $\widehat{T_j}$ [5] is also in T_j, and vice-versa. That is, $Pr(\Sigma_j, \mathbf{T}, \mathbf{S})$ can be computed as

$$Pr(\Sigma_j, \mathbf{T}, \mathbf{S}) = Pr(T_j|\widehat{T_j}) \cdot Pr(\widehat{T_j}|T_j) = Pr(\widehat{T_j}|T_j)^2 \cdot \frac{Pr(T_j)}{Pr(\widehat{T_j})} = Pr(\widehat{T_j}|T_j)^2 \cdot \frac{|T_j|}{|\widehat{T_j}|}. \tag{5}$$

For s source attributes and a fixed j, there are also s possible sets $\Sigma_{j,1}, \ldots, \Sigma_{j,s}$, and $2^s - 1$ non-empty combinations (unions) of them, forming all possible non-trivial sets Σ_j. For computational simplification, we assume that S_{i_1} and S_{i_2} are disjoint for $i_1 \neq i_2$. In other words, each single rule $T_j(d, v) \leftarrow S_{i_k}(d, v)$ with $\Sigma_{j,i_k} \subseteq \Sigma_j$ can be considered in isolation, and we have $\widehat{T_j} = \widehat{T_j}^{\Sigma_j} = \dot{\bigcup}_{i_k} \widehat{T_j}^{\Sigma_{j,i_k}}$.

Thus, instead of $\widehat{T_j}$, we consider $\widehat{T_j}^{\Sigma_{j,i_k}} = S_{i_k}$ (as shown in (3)) and combine the probabilities by summing them up (due to the disjointness assumption):

$$Pr(\widehat{T_j}|T_j) = Pr(\dot{\bigcup_{i_k}} S_{i_k}|T_j) = \sum_k Pr(S_{i_k}|T_j). \tag{6}$$

Thus, in order to compute $Pr(\Sigma_j, \mathbf{T}, \mathbf{S})$, the main task is to compute the $\mathcal{O}(s \cdot t)$ probabilities $Pr(S_i|T_j)$, which we will address in the next section.

In our example with $\Sigma = \Sigma_1 \cup \Sigma_2$, we have to compute:

$$\begin{aligned} Pr(\Sigma, \mathbf{T}, \mathbf{S}) &= Pr(\Sigma_1, \mathbf{T}, \mathbf{S}) \cdot Pr(\Sigma_2, \mathbf{T}, \mathbf{S}) = Pr(\widehat{T_1}|T_1)^2 \cdot \frac{|T_1|}{|\widehat{T_1}|} \cdot Pr(\widehat{T_2}|T_2)^2 \cdot \frac{|T_2|}{|\widehat{T_2}|} \\ &= Pr(\texttt{editor}|\texttt{creator})^2 \cdot \frac{|\texttt{creator}|}{|\widehat{\texttt{creator}}|} \cdot (Pr(\texttt{editor}|\texttt{date}) + Pr(\texttt{created}|\texttt{date}))^2 \\ &\cdot \frac{|\texttt{date}|}{|\widehat{\texttt{date}}|}. \end{aligned}$$

[5] We recall that $\widehat{T_j}$ is obtained from the instance of the source schema by applying the mapping rules. See (3).

3.2. Estimating the probability of a rule

Computing the quality of a mapping is equivalent to estimating the probability $Pr(S_i|T_j)$. Similar to LSD (Doan et al., 2001), the probability $Pr(S_i|T_j)$ is estimated by combining different classifiers $CL_1, \ldots, CL_n$. Each classifier CL_k outputs a conditional probability $Pr(S_i|T_j, CL_k)$, the classifier's approximation of $Pr(S_i|T_j)$. The final probability $Pr(S_i|T_j)$ is computed by combining the classifier predictions $Pr(S_i|T_j, CL_k)$ in a weighted sum, which results from the Total Probability Theorem:

$$Pr(S_i|T_j) \approx \sum_{k=1}^{n} Pr(S_i|T_j, CL_k) \cdot Pr(CL_k). \tag{7}$$

The probabilities $Pr(CL_k)$ are described below.

The classifier predictions are computed in several steps:

(1) First, a classifier CL_k computes a weight $w(S_i, T_j, CL_k)$, an initial approximation of $Pr(S_i|T_j)$.
(2) This weight $w(S_i, T_j, CL_k)$ is then normalized and transformed into the probability $Pr(S_i|T_j, CL_k) = f(w(S_i, T_j, CL_k))$ in two sub-steps:
    (a) First, an arbitrary normalization function can be used, e.g. linear or logistic functions.
    (b) Asecond, simple normalization step ensures that the final value for $Pr(S_i|T_j, CL_k)$ and for $Pr(T_j|S_i, CL_k)$ is in $[0, 1]$.

Step 1, computing the weights $w(S_i, T_j, CL_k)$, is explained in Sections 3.3 and 3.4.

The normalization process is necessary as we combine the classifier estimates, which are heterogeneous in scale. In step 2a, we employ different normalization functions:

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

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 be mapped onto zero, and results above one have to be mapped onto one. The function f_{sum} ensures that the sum of the weights over all source attributes equals 1. Its biggest advantage is that it does not need parameters, which 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 arbitrarily many schema mappings. Of course, normalization functions can be combined. In some cases it might be useful to bring the classifier weights into the same range (using f_{sum}), and the 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

$$0 \leqslant Pr(S_i|T_j, CL_k) \leqslant \frac{\min(|S_i|, |T_j|)}{|T_j|} = \min\left(\frac{|S_i|}{|T_j|}, 1\right). \tag{8}$$

Thus, the normalized value (which is in $[0, 1]$) is multiplied with $\min(|S_i|/|T_j|, 1)$ in a second normalization step (step 2b).

Finally, 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 can simply use $Pr(CL_k) = \frac{1}{n}$ for $1 \leqslant k \leqslant n$, i.e. the predictions are averaged. As an alternative, $Pr(CL_k)$ is considered as the probability that CL_k is the best classifier. With the mean error

$$E(S_i, t, CL_k) = (Pr(S_i|\{t\}, CL_k) - Pr(S_i|\{t\}))^2,$$

the probability can be computed as

$$Pr(CL_k) = \frac{\sum_j \sum_i |\{t | t \in T_j^J, CL_k = \operatorname{argmin} E(S_i, t, CL_{k'})\}|}{s \cdot \sum_j |T_j^J|}. \tag{9}$$

3.3. Schema-based classifiers

The following classifiers consider the schemas only, without knowledge about instances (the latter are described in Section 3.4).

3.3.1. Same attribute names

This binary classifier CL_N returns a weight of 1 if and only if the two attributes have the same name, and 0 otherwise:

$$w(S_i, T_j, CL_N) = \begin{cases} 1, & S_i = T_j, \\ 0, & \text{otherwise.} \end{cases}$$

3.3.2. Same attribute name stems

This binary classifier CL_S uses the stemming technique well known from information retrieval and natural-language processing. Here, a word is mapped onto its "stem" to provide problems with derivation forms.

For CL_S, stemming is applied onto attribute names. The classifier returns a weight of 1 if and only if the names of the two attributes have the same stem (using e.g. a Porter stemmer), and 0 otherwise:

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

3.3.3. Same data type

This binary classifier CL_D returns a weight of 1 if and only if the two attributes use the same data type, and 0 otherwise.

$$w(S_i, T_j, CL_D) = \begin{cases} 1, & d_{S_i} = d_{T_j}, \\ 0, & \text{otherwise.} \end{cases}$$

E.g., we have $w(\texttt{editor}, \texttt{date}, CL_D) = 0$ as `editor` and `date` have different data types, and $w(\texttt{created}, \texttt{date}, CL_D) = 1$ as both have the same data type `Date`.

3.4. Content-based classifiers

These classifiers are based on attribute values, and thus require instances of both schemas. However, these instances do not need to describe the same objects. Below, we describe the classifiers used in this paper.

Some of these classifiers require term weights inside strings. We consider strings as bags of words, and then can use normalized term frequencies as conditional probabilities that an arbitrary word in the text v equals a specified word w:

$$Pr(w|v) = \frac{tf(w, v)}{\sum_{w' \in v} tf(w', v)},$$

Here, $tf(w, v)$ denotes the number of times the word w appears in the string v, which is normalized by the sum of the frequencies of all words in v.

3.4.1. Correct literals

This classifier CL_L (suitable in particular for numbers, URLs and other factual data) measures the fraction of the tuples in T_j where the data value (the second argument, without the document id) also occurs in any tuple in S_i:

$$w(S_i, T_j, CL_L) = \frac{|\{T_j(d_1, v_1) : T_j(d_1, v_1) \in T_j, \exists S_i(d_2, v_2) \in S_i.v_1 = v_2\}|}{|T_j|}.$$

This directly corresponds to the conditional probability $Pr(S_i|T_j)$, but is unable to cope with partial or imprecise information.

3.4.2. kNN classifier

A popular classifier for text and factual data is kNN (Sebastiani, 2002). For CL_{kNN}, each attribute S_i acts as a category, and training sets are formed from the instances of S_i by labelling each document/value pair with the schema attribute S_i:

$$Train = \bigcup_{i=1}^{s} \{(S_i, d', v') : S_i(d', v') \in S_i\}.$$

For instance, given a set of documents in the `BIBDB` source schema, then e.g. the `journal` attribute acts as a category and the set of journal names is the training set for the classifier.

For every instance $T_j(d, v) \in T_j$, e.g. for each value of the `booktitle` attribute in the `standard` target schema, the k-nearest neighbours TOP_k have to be found by ranking the values $(S_i, d', v') \in Train$ according to their similarity $RSV(v, v')$. The prediction weights are then computed by summing up the similarity values for all (d', v') which are built from S_i, and by averaging these weights $\tilde{w}(d, v, S_i)$ over all $T_j(d, v) \in T_j$:

$$w(S_i, T_j, CL_{kNN}) = \frac{1}{|T_j|} \cdot \sum_{T_j(d,v) \in T_j} \tilde{w}(d, v, S_i),$$

$$\tilde{w}(d, v, S_i) = \sum_{(S_l, d' v') \in TOP_k, S_i = S_l} RSV(v, v'),$$

$$RSV(v, v') = \sum_{w \in v \cap v'} Pr(w|v) \cdot Pr(w|v').$$

3.4.3. Naive Bayes text classifier

The classifier CL_B uses a naive Bayes text classifier (Sebastiani, 2002) for text content. Again, each attribute acts as a category, and attribute values are considered as bags of words (with normalized word frequencies as probability estimations, see above). For each $T_j(d, v) \in T_j$, 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

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

With $Pr(S_i)$ we denote the probability that a randomly chosen value in $\cup_k S_k$ is a value in S_i, and $Pr(w|S_i) = Pr(w|v(S_i))$ is defined as for kNN, where $v(S_i) = \bigcup_{S_i(d,v) \in S_i} v$ is the combination of all words in all values for all objects in S_i. 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

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

In the case that 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.

3.4.4. KL-distance

The classifier CL_{KL} uses KL-distance to measure the distance between the distributions of numerical, categorical and textual data. In general, KL-distance measures how good one distribution p approximates another distribution q, and is defined as

$$KL(p||q) = \sum_{x \in X} p(x) \cdot \log \frac{p(x)}{q(x)}.$$

For non-textual data, we only consider the attribute values, i.e. we consider the projections $S'_i = \{v | \exists d : S_i(d, v) \in S_i\}$ and $T'_j = \{v | \exists d : T_j(d, v) \in T_j\}$. For textual data, the text is split into terms, and attribute-wide statistics are used for estimating the probabilities $Pr(X = Z)$ and $Pr(Y = z)$, where z is a term.

The KL-distance distance $KL(S'_i||T'_j) \in [0, \infty$ is converted into a prediction by:

$$w(S_i, T_j, CL_{KL}) = \frac{1}{1 + KL(S'_i||T'_j)}.$$

3.5. Estimating the weight of a rule

The learned schema mapping is a set of deterministic rules, i.e. no probabilistic weights are attached. The final rules weights $Pr(T_j|S_i)$ for these rules have to be computed in a post-processing step. Two different approaches are described in this section.

3.5.1. Rule (ST) – transforming the probability $Pr(S_i|T_j)$

The probability $Pr(S_i|T_j)$ has already been estimated while computing the quality of a schema mapping. This probability can be easily transformed in the rule weight:

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

As the final normalization step in Section 3.2 ensures that $Pr(S_i|T_j) \leqslant \min(|S_i|/|T_j|, 1)$ (see Eq. (8)), the resulting value $Pr(T_j|S_i)$ is always in $[0, 1]$.

3.5.2. Rule(TS) – Estimating the probability $Pr(T_j|S_i)$ directly

The probability $Pr(T_j|S_i)$ can be learned just as the probability $Pr(S_i|T_j)$ (see Section 3.2) by using a set of classifiers. The same ideas, the same classifiers and the same normalization functions can be used as described above, swapping S_i and T_j.

3.6. Additional constraints

Additional constraints can be applied on the learned rules for improved precision. These constraints are used after the sets of rules are learned for all target classes: we remove learned rules, which violate one of these constraints:

(1) We can assume that there is at most one rule for the target class (the one with the highest weight). This will reduce the number of rules produced, and hopefully increase the percentage of correct rules.
(2) We can drop all rules where the weight $\alpha_{j,i}$ is lower than a threshold ε. We consider $\varepsilon = 0.1$.
(3) We can rank the rules according to their weights (in decreasing order), and use the n top-ranked rules (e.g. about the actual number of rules).

4. Experiments

This chapter describes some experiments for evaluating the presented learning approach. It also introduces the usual effectiveness measurements for evaluating the learned schema mappings (precision, recall

Table 1
Example of BIBDB and standard schema entry, respectively

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<document id="BIBDB.r0/1">
<doctype>article</doctype>
<author>Imielinski, T.</author>
<title>Incomplete deductive databases</title>
<journal>Annals of Mathematics and Artificial Intelligence</journal>
<volume>3</volume>
<number>2-4</number>
<pages>259-294</pages>
<year>1991</year>
</document>

<?xml version="1.0" encoding="ISO-8859-1" ?>
<document id="BIBDB.r0/9">
<author>Andersen, T.L. and Ecklund, E.F. and Maier, D.</author>
<year>1988</year>
<type>article</type>
<title>The Proteus Bibliography: Representation and Interactive Display in Databases</title>
<booktitle>ACM SIGMOD Record</booktitle>
<misc>Volume: 15</misc>
<misc>Number: 3</misc>
<pages>46-55</pages>
</document>
```

and F-measure). Furthermore, we address the issue of evaluating each classifier individually, to determine their impact on effectiveness. To the best of our knowledge, this has not yet been addressed in previous work.

4.1. Evaluation setup

This section describes the test set (source and target instances) and the classifiers used for the experiments. Experiments were performed on four test beds, whose data contain mainly textual data (see Tables 1–3 for example entries)

- BIBDB contains over 3000 BibTeX entries about information retrieval and related areas. The entries are available both in BibTeX (source schema) and in a manually created standard schema, derived from BibTeX via simple rules. Both schemas share a large amount of common attribute names. The target contains 26 attributes, the source contains 11 attributes and there are 11 correct mapping rules to be discovered. There are 9 target attributes for which there is at least one mapping.
- NGA is a sampled collection of 864 entries from the National Gallery of Arts, Washington, DC. The data is available both in a source schema (manually constructed from the Web site) and in the same standard schema also used for BIBDB. The target contains 26 attributes, the source contains 16 attributes and there are 16 correct rules to be discovered. There are 9 target attributes for which there is at least one mapping.
- WebArt is a sampled collection of 265 entries from the Web Gallery of Art. The entries are available both in a source schema (manually constructed from the Web site) and in same standard schema also used for BIBDB and NGA. The target contains 26 attributes, the source contains 11 attributes and there are 11 correct rules to be discovered. There are 10 target attributes for which there is at least one mapping.
- LOC is an Open Archive collection of the Library of Congress with about 1,700 entries, available in MARC 21 (source schema) and in Dublin Core (target schema). MARC 21 has a higher granularity as Dublin Core, a lot of Dublin Core attribute values are the concatenation of several MARC 21 attributes. Both schemas use a completely different name scheme, thus they do not have attribute names in common. The target contains 10 attributes (Dublin Core), the source contains 31 attributes and there are 43 correct rules to be discovered. There are 8 target attributes for which there is at least one mapping.

Table 2
Example of NGA and standard scheme entry, respectively

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<document id="NGA//cgi-bin/pinfo?Object=14121+0+none">
<image-url>http://www.nga.gov/image/a00003/a0000352.jpg</image-url>
<image-width>610</image-width>
<image-height>324</image-height>
<artist>Marie Alain</artist>
<form>watercolor, graphite, and gouache on paperboard, 37.6 x 50.9 cm (14 13/16 × 20 1/16 in.)</form>
<accession-number>1943.8.1907</accession-number>
<info>These brocaded silk shoes exemplify the beautiful and elaborate accessories that were
part of the fashionable eighteenth-century wardrobe. Since they were not suitable for walking
out-of-doors, clogs or pattens — separate wooden or leather soles — were strapped on over
the shoes when necessary.</info>
<bibliography>Christensen, Erwin O.,The Index of American Design, New York: 1950, p. 177, no. 348.</
bibliography>
</document>

<?xml version="1.0" encoding="ISO-8859-1" ?>
<document id="NGA//cgi-bin/pinfo?Object=60868+0+none">
<image-url>http://www.nga.gov/image/a00022/a00022a1.jpg</image-url>
<image-width>275</image-width>
<image-height>390</image-height>
<author>Antonio Balestra</author>
<title>Venus Appearing to Aeneas</title>
<yearperiod>in or before 1725</yearperiod>
<type>pen and brown ink with brown wash over black chalk on laid paper, 23.3 × 16.2 cm (9 1/8 × 6 3/8 in.)</
type>
<misc>Accession-number: 1982.17.2</misc>
<misc>Bibliography: The Glory of Venice. Exh. cat. Royal Academy of Arts, London; National Gallery of Art,
Washington; Museo del Settecento Veneziano - Ca'Rezzonico, Venice; Gallerie dell'Accademia, Venice,
1994-1995: no. 14.</misc>
<misc>Provenance: J. McGowan (Lugt 1496)(sale, 1804); William Esdaile. Private collection (sale, Chris-
tie's, London, 8 December 1981, no. 64); (David Tunick, New York); NGA purchase in 1982.</misc>
</document>
```

The collections BIBDB, NGA and WebArt are created by the MIND project,[6] while LOC has been harvested within the Cyclades project.[7]

Each of the collections BIBDB, NGA, WebArt and LOC 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 schemas). 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).

For the BIBDB, NGA and WebArt collections, each of the classifiers introduced in Sections 3.3 and 3.4 are used alone. In addition, the combination of the schema-based classifiers CL_N, CL_S and CL_D is used, as well as the combination of the content-oriented classifiers CL_{kNN}, CL_B, CL_L and CL_{KL}, and the combination of all classifiers. In contrast, the schema-oriented classifiers do not help for the LOC collection (no attribute names in common, and always same data type `Text`). Thus, only the content-oriented classifiers (and their combination) are used for this collection.

4.2. Measures

We evaluate if the learning approach computes the correct rules (neglecting the corresponding rule weights). Similar to the area of Information Retrieval, precision defines how many learned rules are correct,

[6] http://www.mind-project.org.
[7] http://www.ercim.org/cyclades/.

Table 3
Example of MARC and DC entry, respectively

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<oai_marc>
<varfield id="100">
<subfield label="a">Hondius, Jodocus,</subfield> </varfield>
<varfield id="100">
<subfield label="d">1563-1612.</subfield>
</varfield>
<varfield id="245">
<subfield label="a">America noviter delineata /</subfield>
</varfield>
<varfield id="245">
<subfield label="c">auct. Jodoco Hondio ; H. Picart fecit.</subfield>
</varfield>
<varfield id="260">
<subfield label="a">[Paris :</subfield>
</varfield>
<varfield id="260">
<subfield label="b">Jean Boisseau,</subfield>
</varfield>
<varfield id="260">
<subfield label="c">1640?]</subfield>
</varfield>
<varfield id="300">
<subfield label="a">1 map ;</subfield>
</varfield>
<varfield id="500">
<subfield label="a">Relief shown pictorially.</subfield>
</varfield>
<varfield id="500">
<subfield label="a">Includes insets of north and south polar regions.</subfield>
</varfield>
<varfield id="650">
<subfield label="a">Maps, Early.</subfield>
</varfield>
<varfield id="700">
<subfield label="a">Picart, H.</subfield>
</varfield>
<varfield id="700">
<subfield label="a">Boisseau, Jean,</subfield>
</varfield>
<varfield id="700">
<subfield label="d">1587-1662 or 4.</subfield>
</varfield>
<varfield id="700">
<subfield label="d">fl. 1637-1658.</subfield>
</varfield>
<varfield id="700">
<subfield label="q">(Hugues),</subfield>
</varfield>
<varfield id="856">
<subfield label="u">http://hdl.loc.gov/loc.gmd/g3290.np000144</subfield>
</varfield>
</oai_marc>
```

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<dc>
<coverage>Charleston (S.C.)</coverage>
<coverage>New York (N.Y.)</coverage>
<coverage>United States—South Carolina—Charleston.</coverage>
```

(*continued on next page*)

Table 3 (*continued*)

```
<coverage>United States—New York (State)—New York.</coverage>
<date>1762-1776</date>
<description>Charleston view is panorama of waterfront and quays with principal buildings lettered for
identification (no index present).</description>
<description>Relief shown by hachures on New York City map.</description>
<description>New York City map removed from London Magazine, Sept. 1776; Charleston view removed from
London Magazine, 1762.</description>
<description>Imperfect: Charleston view ink-stained at upper left. DLC</description>
<description>LC Maps of North America, 1750-1789, 1111</description>
<description>New York City map includes index to points of interest.</description>
<description>New York City map shows streets, wharves, ferries, principal buildings, and built-up area.</
description>
<identifier>http://hdl.loc.gov/loc.gmd/ar3804n.ar111100</identifier>
<language>eng</language>
<publisher>[London :]</publisher>
<title>Plan of the city of New York ; An exact prospect of Charlestown : the metropolis of the province of
South Carolina.</title>
<type>image</type>
<type>map</type>
<type>cartographic</type>
</dc>
```

and recall defines how many correct rules are learned. Finally, the F-measure denotes the harmonic mean of precision and recall. 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. Then:

$$precision = \frac{|R_L \cap R_A|}{|R_L|}, \quad recall = \frac{|R_L \cap R_A|}{|R_A|}, \quad F = \frac{2}{\frac{1}{precision} + \frac{1}{recall}}.$$

Finally, we also used a variant of traditional precision (called "restricted 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.

4.3. Results

In the experiments presented in this section (Tables 4–8), the learning steps are as follows:

(1) Find the best schema mapping
  (a) Estimate the probabilities $Pr(S_i|T_j)$ for all source attributes $S_i \in \mathbf{S}$ and all target attributes $T_j \in \mathbf{T}$, using all classifiers.
  (b) For every target relation T_j and for every non-empty subset Σ_j of schema mapping rules having T_j as head, estimate the probability $Pr(\Sigma_j, \mathbf{T}, \mathbf{S})$.
  (c) Select the rule set Σ_j, which maximizes the probability $Pr(\Sigma_j, \mathbf{T}, \mathbf{S})$.
(2) Estimate the weights $Pr(T_j|S_i)$ for the learned rules by converting $Pr(S_i|T_j)$ or learn them directly.
(3) Compute the precision and recall as described above.

4.3.1. Quality of the classifiers with $Pr(CL_k) = 1/n$

For the BIBDB collection, precision of 1.0 is obtained by CL_N and CL_S for this rather simple collection, followed by CL_L (0.875) and the combination of all classifiers (0.7155). Recall is maximized by the $CL_N + CL_S + CL_D$ (0.9318) and CL_D (slightly worse). The best content-oriented classifier is CL_{KL} (whose precision is worst), Naive Bayes performs worst. The ranking for the F-measure is comparable to the precision; and restricted precision performs about the same as traditional precision.

Table 4
ST-Rule (ST), no constraints – BIBDB

	f_{id}	f_{sum}	$f_{lin} \circ f_{sum}$	$f_{log} \circ f_{sum}$
(a) Precision				
CL_N	1.0000	1.0000	1.0000	1.0000
CL_S	1.0000	1.0000	1.0000	1.0000
CL_D	0.2632	0.2632	0.2632	0.2632
CL_{kNN}	0.5455	0.5455	0.5455	0.5000
CL_B	0.5000	0.5000	0.5000	0.5000
CL_L	0.8750	0.8750	0.8750	0.8571
CL_{KL}	0.2353	0.2353	0.2353	0.2286
All schema	0.2973	0.4348	0.4348	0.4348
All content	0.3462	0.4211	0.4211	0.4444
All	0.4074	0.8182	0.8182	0.8182
(b) Recall				
CL_N	0.6364	0.6364	0.6364	0.6364
CL_S	0.6364	0.6364	0.6364	0.6364
CL_D	0.9091	0.9091	0.9091	0.9091
CL_{kNN}	0.5455	0.5455	0.5455	0.3636
CL_B	0.4545	0.4545	0.4545	0.3636
CL_L	0.6364	0.6364	0.6364	0.5455
CL_{KL}	0.7273	0.7273	0.7273	0.7273
All schema	1.0000	0.9091	0.9091	0.9091
All content	0.8182	0.7273	0.7273	0.7273
All	1.0000	0.8182	0.8182	0.8182
(c) F-measure				
CL_N	0.7778	0.7778	0.7778	0.7778
CL_S	0.7778	0.7778	0.7778	0.7778
CL_D	0.4082	0.4082	0.4082	0.4082
CL_{kNN}	0.5455	0.5455	0.5455	0.4211
CL_B	0.4762	0.4762	0.4762	0.4211
CL_L	0.7368	0.7368	0.7368	0.6667
CL_{KL}	0.3556	0.3556	0.3556	0.3478
All schema	0.4583	0.5882	0.5882	0.5882
All content	0.4865	0.5333	0.5333	0.5517
All	0.5789	0.8182	0.8182	0.8182

Optimum precision of 1.0 is obtained for NGA by CL_N and CL_S (as both schemas share a large amount of attribute names), but also by Naive Bayes and CL_L. Recall is maximized by the combination of the content-oriented classifiers and KL-distance alone (both 0.4375), the name-based classifiers perform (not surprisingly) worst with only 0.0625. The combination of all classifiers obtain the best F-measure (0.4799 on average), directly followed by the combination of all content-oriented classifiers and kNN (5% worse).

For WebArt, precision of 1.0 can be obtained by the same classifiers as for NGA. The combination of classifiers, the combination of all content-oriented classifiers and CL_L all lead to a recall level of 0.5455, followed by KL-distance (17% worse). F-measure, again is optimized by CL_L and the combination of all classifiers.

For LOC, we only considered content-oriented classifiers, as the schemas are completely different in nature, do not share any common name, and do not use different data types. The DC attributes (in the target instances) often contain the concatenation of several MARC attributes. Precision is maximized by CL_L (if values are the same, then there is a match) with 0.7250, followed by kNN (25% worse). For recall, again CL_{KL} yields the best quality (0.2927, showing the difficulty of this collection), and CL_B is the worst classifier. kNN and CL_L also maximize the F-measure, while KL-distance performs poor.

Average precision (over all collections, classifiers etc.) is 0.6024, average restricted precision is slightly higher with 0.6134. This means that there are almost only rules for target attribute which actually participate in any mapping. The only exception is the LOC collection, where average precision is 0.4295 and average restricted precision is 0.5186.

Table 5
ST-Rule (ST), no constraints – NGA

	f_{ids}	f_{sum}	$f_{lin} \circ f_{sum}$	$f_{log} \circ f_{sum}$
(a) Precision				
CL_N	1.0000	1.0000	1.0000	1.0000
CL_S	1.0000	1.0000	1.0000	1.0000
CL_D	0.2400	0.2400	0.2400	0.2400
CL_{kNN}	0.8333	0.8333	0.8333	0.8333
CL_B	1.0000	1.0000	1.0000	1.0000
CL_L	1.0000	1.0000	1.0000	1.0000
CL_{KL}	0.1795	0.1795	0.1795	0.1842
All schema	0.2400	0.3000	0.3000	0.3000
All content	0.1795	0.7000	0.7000	0.8750
All	0.1842	0.7000	0.7000	0.8750
(b) Recall				
CL_N	0.0625	0.0625	0.0625	0.0625
CL_S	0.0625	0.0625	0.0625	0.0625
CL_D	0.3750	0.3750	0.3750	0.3750
CL_{kNN}	0.3125	0.3125	0.3125	0.3125
CL_B	0.2500	0.2500	0.2500	0.2500
CL_L	0.2500	0.2500	0.2500	0.2500
CL_{KL}	0.4375	0.4375	0.4375	0.4375
All schema	0.3750	0.3750	0.3750	0.3750
All content	0.4375	0.4375	0.4375	0.4375
All	0.4375	0.4375	0.4375	0.4375
(c) F-measure				
CL_N	0.1176	0.1176	0.1176	0.1176
CL_S	0.1176	0.1176	0.1176	0.1176
CL_D	0.2927	0.2927	0.2927	0.2927
CL_{kNN}	0.4545	0.4545	0.4545	0.4545
CL_B	0.4000	0.4000	0.4000	0.4000
CL_L	0.4000	0.4000	0.4000	0.4000
CL_{KL}	0.2545	0.2545	0.2545	0.2593
All schema	0.2927	0.3333	0.3333	0.3333
All content	0.2545	0.5385	0.5385	0.5833
All	0.2593	0.5385	0.5385	0.5833

4.3.2. *Quality of the normalization functions with $Pr(CL_k) = 1/n$*

The tables show the importance of using the best normalization function, as the results differ dramatically in some cases: E.g., for BIBDB and the combination of all classifiers precision for the identity function is 0.4074, while is it about twice as high for the other normalization functions.

There is no consistent best normalization function. Considering precision, $f_{log} \circ f_{sum}$ performs best on average, and the identity function is the worst one. Recall is maximized by the identity function and $f_{log} \circ f_{sum}$. Finally, $f_{log} \circ f_{sum}$ and f_{sum} yield the highest F-measure.

4.3.3. *Runs with learned $Pr(CL_k)$*

In the experimental runs presented so far, the average of the classifier predictions has been computed. We also conducted experiments where the combination weights $Pr(CL_k)$ are learned as described in Section 3.2, Eq. (9). Of course learned probabilities only play a role when at least two classifiers are combined.

The differences to the case with averaged predictions are very small (smaller than 5% in all cases). Precision, recall and F-measure are higher for the run with learned probabilities.

4.3.4. *Runs with constraints*

Table 8 shows the results for the different constraints. Experiments have been conducted without any constraint, will all constraints described in Section 3.6, and with all three constraints combined.

Table 6
ST-Rule (ST), no constraints – WebArt

	f_{id}	f_{sum}	$f_{lin} \circ f_{sum}$	$f_{log} \circ f_{sum}$
(a) Precision				
CL_N	1.0000	1.0000	1.0000	1.0000
CL_S	1.0000	1.0000	1.0000	1.0000
CL_D	0.2381	0.2381	0.2381	0.2381
CL_{kNN}	1.0000	1.0000	1.0000	1.0000
CL_B	1.0000	1.0000	1.0000	1.0000
CL_L	1.0000	1.0000	1.0000	1.0000
CL_{KL}	0.2941	0.2941	0.2941	0.2381
All schema	0.2941	0.3846	0.3846	0.3846
All content	0.3333	1.0000	1.0000	1.0000
All	0.4615	1.0000	1.0000	1.0000
(b) Recall				
CL_N	0.2727	0.2727	0.2727	0.2727
CL_S	0.2727	0.2727	0.2727	0.2727
CL_D	0.4545	0.4545	0.4545	0.4545
CL_{kNN}	0.4545	0.4545	0.4545	0.4545
CL_B	0.4545	0.4545	0.4545	0.4545
CL_L	0.5455	0.5455	0.5455	0.5455
CL_{KL}	0.4545	0.4545	0.4545	0.4545
All schema	0.4545	0.4545	0.4545	0.4545
All content	0.5455	0.5455	0.5455	0.5455
All	0.5455	0.5455	0.5455	0.5455
(c) F-measure				
CL_N	0.4286	0.4286	0.4286	0.4286
CL_S	0.4286	0.4286	0.4286	0.4286
CL_D	0.3125	0.3125	0.3125	0.3125
CL_{kNN}	0.6250	0.6250	0.6250	0.6250
CL_B	0.6250	0.6250	0.6250	0.6250
CL_L	0.7059	0.7059	0.7059	0.7059
CL_{KL}	0.3571	0.3571	0.3571	0.3125
All schema	0.3571	0.4167	0.4167	0.4167
All content	0.4138	0.7059	0.7059	0.7059
All	0.5000	0.7059	0.7059	0.7059

Table 7
ST-Rule (ST), no constraints – LOC

	f_{id}	f_{sum}	$f_{lin} \circ f_{sum}$	$f_{log} \circ f_{sum}$
(a) Precision				
CL_{kNN}	0.6667	0.6667	0.6667	0.1791
CL_B	0.4375	0.4375	0.4375	0.1731
CL_L	0.8000	0.8000	0.8000	0.5000
CL_{KL}	0.1714	0.1714	0.1714	0.1765
All content	0.2031	0.6429	0.6429	0.2292
(b) Recall				
CL_{kNN}	0.2439	0.2439	0.2439	0.2927
CL_B	0.1707	0.1707	0.1707	0.2195
CL_L	0.1951	0.1951	0.1951	0.2683
CL_{KL}	0.2927	0.2927	0.2927	0.2927
All content	0.3171	0.2195	0.2195	0.2683
(c) F-measure				
CL_{kNN}	0.3571	0.3571	0.3571	0.2222
CL_B	0.2456	0.2456	0.2456	0.1935
CL_L	0.3137	0.3137	0.3137	0.3492
CL_{KL}	0.2162	0.2162	0.2162	0.2202
All content	0.2476	0.3273	0.3273	0.2472

Table 8
Precision, recall and F-Measure without and with constraints

	Precision	Recall	F-Measure
(a) BIBDB			
No constraints	0.6288	0.6364	0.5733
Constraint 1	0.6779	0.4351	0.5299
Constraint 2	0.6985	0.5649	0.5631
Constraint 3	0.6098	0.4545	0.5165
All constraints	0.7060	0.4026	0.5077
(b) NGA			
No constraints	0.7506	0.2500	0.2912
Constraint 1	0.7905	0.1607	0.2555
Constraint 2	0.8265	0.2054	0.2639
Constraint 3	0.7552	0.1987	0.2776
All constraints	0.8476	0.1429	0.2331
(c) WebArt			
No constraints	0.7579	0.4455	0.5161
Constraint 1	0.8200	0.3636	0.4975
Constraint 2	0.8188	0.4023	0.4954
Constraint 3	0.7568	0.3886	0.4940
All constraints	0.8550	0.3477	0.4817
(d) LOC			
No constraints	0.4487	0.2402	0.2766
Constraint 1	0.6600	0.1524	0.2473
Constraint 2	0.4306	0.2000	0.2450
Constraint 3	0.4551	0.2110	0.2698
All constraints	0.6521	0.1366	0.2235

As expected, precision can be dramatically improved for all five collections when using at least one of the constraints. For BIBDB, NGA and WebArt, applying all three constraints yield the best result, for LOC, constraint 1 alone ("at most one rule per target attribute") performs best. In contrast, recall is dramatically hurt when using any constraint; the same holds for F-measure.

4.3.5. Conclusion

No classifier, normalization functions or constraints consistently outperform all others, so the concrete choice depends on the collection and the goal. E.g., some classifiers obtain high precision (by returning a low number of high-confident rules), others yield high recall (by returning a large amount of mapping rules). E.g., CL_{KL} yields high recall but low precision. As the classifiers follow one of these extremes in most cases, it is difficult to find a trade-off between precision and recall. In general, however, precision and recall is quite high (values above 0.8 can be obtained), with the exception of recall for LOC (a difficult collection).

Name-based classifiers typically provide excellent precision and rather low recall (depending on the schemas). In addition, content-oriented classifiers proved to be very useful for schema matching (w.r.t. both precision and recall). This is important in general settings where schema names alone are not sufficient to find matching attributes, as e.g. for DC vs. MARC. The experiments further showed that using a normalization function, combining classifiers and applying constraints can significantly increase the matching quality. In contrast, using classifiers also for learning the rule weights and learning the classifier combination weights does not lead to increased quality.

5. Related work

Our model sPLMap has its foundations in two related research areas, *schema matching* (Rahm & Bernstein, 2001) and *information exchange* (Fagin et al., 2003), and borrows from them the terminology and ideas. Related to the latter, we view the matching problem as the problem of determining the "best possible set Σ" such that the *exchange* of instances of a source class into a target class has the highest probability of being correct. From the former we inherit the requirement to rely on machine learning techniques to automate the process of schema matching. In terms of these frameworks, sPLMap is built upon the GLaV (global-and-local-as-view) approach, where schemas are built independently (no schema is constructed as a view of the other).

Several different approaches for the schema matching problem have been proposed within the last few years, e.g. (Bilke & Neumann, 2005; Dhamankar, Lee, Doan, Halevy, & Domingos, 2004; Do & Rahm, 2002; Doan, Domingos, & Halevy, 2003; Doan et al., 2001; Embley, Jackman, & Xu, 2001; Fagin et al., 2003; Kang & Naughton, 2003; Madhavan, Bernstein, Chen, & Halevy, 2005; Melnik, Garcia-Molina, & Rahm, 2002) (see (Dhamankar et al., 2004; Rahm & Bernstein, 2001) for a more extensive comparison). Most recent approaches either implicitly or explicitly perform the schema mapping based on attribute name comparison, comparing properties of the underlying data instances and the structure in which an attribute is embedded.

Some approaches like (Bilke & Neumann, 2005; Kang & Naughton, 2003; Madhavan et al., 2005; Melnik et al., 2002) (to mention a few) provide either implicitly or explicitly an estimate of the quality of a set of mappings and then try to maximize this quality based on e.g. weighted graph matching theory. For instance, informally, in Kang and Naughton (2003) the estimate is a distance between the weighted target schema dependency graph and the weighted source schema dependency graph, while in e.g. (Bilke & Neumann, 2005) the estimate is the ‘cost’ of a bipartite weighted matching.

To the best of our knowledge, none of the other approaches combine in a unique formal framework logic with statistical learning and heuristics. We believe that this latter aspect is of particular importance as it constitutes the base to extend our model to cases with more expressive rules (e.g. (Fagin, Kolaitis, Tan, & Popa, 2004)), schema languages (e.g. (Calvanese, Lenzerini, & Nardi, 1998)) or schemas (e.g. ontology description languages like OWL and OWL DL (Horrocks, Patel-Schneider, & van Harmelen, 2003)). As a consequence, all aspects of logical reasoning, considered as important, can be plugged into our model.

6. Conclusion and outlook

With the proliferation of data sharing applications over the Web, the development of automated tools for schema matching will be of particular importance. In this paper, we have presented a Probabilistic, Logic-based formal framework for Schema Matching, which for ease of presentation we call *sPLMap*. The peculiarity of our approach is that it neatly combines machine learning, information retrieval and heuristic techniques for learning a set of mapping rules. An important application area is distributed IR, where it is neither feasible to impose a single schema nor to transform all documents into a standard schema in a pre-processing step. Instead, queries and documents have to be transformed on-the-fly during runtime.

As future work, we see some appealing points. Fitting new classifiers into our model is straightforward theoretically, but it is quite more difficult to practically find the most appropriate one or a combination of them. In the future, more variants will be embedded, developed and evaluated to improve the quality of the learning mechanism.

Probabilistic Datalog allows for extended rules, which e.g. combine the content of several attributes, or perform a data type conversion (Nottelmann & Straccia, 2004; Nottelmann & Straccia, 2005): For this, we introduce (probabilistic) binary operators, which compare values from two (potentially different) data types. Then, an additional conversion predicate is required which allows for transforming data type values and operator results between data types (and, potentially, different operators).

An interesting research direction is to evaluate the combination of multiple *methods* existing in the literature. Our framework could be extended in a straightforward way. Like we have done for classifiers, we can extend $Pr(\Sigma, \mathbf{T}, \mathbf{S})$ by combining these methods: Each method M_u computes a weight $w(\Sigma, \mathbf{S}, \mathbf{T}, M_u)$, which is the method's initial approximation of $Pr(\Sigma, \mathbf{T}, \mathbf{S})$. This weight $w(\Sigma, \mathbf{S}, \mathbf{T}, M_u)$ will then be normalized and transformed into a probability $Pr(\Sigma, \mathbf{T}, \mathbf{S}|M_u) = f(w(\Sigma, \mathbf{S}, \mathbf{T}, M_u))$, the method's estimation of $Pr(\Sigma, \mathbf{T}, \mathbf{S})$. All the estimations $Pr(\Sigma, \mathbf{T}, \mathbf{S}|M_u)$ will then be combined together as

$$Pr(\Sigma, \mathbf{T}, \mathbf{S}) = \sum_{u=1}^{m} Pr(\Sigma, \mathbf{T}, \mathbf{S}|M_u) \cdot Pr(M_u),$$

where $Pr(M_u)$ is a measure indicating the effectiveness of the method M_u. In that case, the method proposed in this paper may be just one of the methods $M_1, \ldots, M_m$. Optimization, i.e. determining the set Σ with the highest quality, will then be performed 'globally'.

Acknowledgements

This work is supported in part by the DFG (grant BIB47 DOuv 02-01, project "Pepper") and in part by ISTI-CNR (project "Distributed Search in the Semantic Web").

References

Bilke, A., & Neumann, F. (2005). Schema matching using duplicates. In *Proceedings of the 21st international conference on data engineering (ICDE-05)* (pp. 69–80). IEEE Computer Society.

Calvanese, D., Lenzerini, M., & Nardi, D. (1998). Description logics for conceptual data modeling. In *Logics for databases and information systems* (pp. 229–263).

Dhamankar, R., Lee, Y., Doan, A., Halevy, A., & Domingos, P. (2004). iMAP: discovering complex semantic matches between database schemas. In *Proceedings of the 2004 ACM SIGMOD international conference on management of data* (pp. 383–394). ACM Press.

Doan, A., Domingos, P., & Halevy, A. Y. (2001). Reconciling schemas of disparate data sources: a machine-learning approach. In *Proceedings of the 2001 ACM SIGMOD international conference on management of data* (pp. 509–520). ACM Press.

Doan, A., Domingos, P., & Halevy, A. (2003). Learning to match the schemas of data sources: a multistrategy approach. *Machine Learning, 50*(3), 279–301.

Do, H., & Rahm, E. (2002). Coma – a system for flexible combination of schema matching approaches. In *Proceedings of the international conference on very large data bases (VLDB-02)*.

Embley, D. W., Jackman, D., & Xu, L. (2001). Multifaceted exploitation of metadata for attribute match discovery in information integration. In *Workshop on information integration on the Web* (pp. 110–117).

Fagin, R., Kolaitis, P. G., Tan, W.-C., & Popa, L. (2004). Composing schema mappings: second-order dependencies to the rescue. In *Proceedings PODS*.

Fagin, R., Kolaitis, P. G., Miller, R., & Popa, L. (2003). Data exchange: semantics and query answering. In *Proceedings of the international conference on database theory (ICDT-03). Lecture notes in computer science* (Vol. 2572, pp. 207–224). Springer-Verlag.

Fuhr, N. (2000). Probabilistic datalog: implementing logical information retrieval for advanced applications. *Journal of the American Society for Information Science, 51*(2), 95–110.

Horrocks, I., Patel-Schneider, P. F., & van Harmelen, F. (2003). From SHIQ and RDF to OWL: the making of a Web ontology language. *Journal of Web Semantics, 1*(1), 7–26.

Kang, J., & Naughton, J. F. (2003). On schema matching with opaque column names and data values. In *Proceedings of the 2003 ACM SIGMOD international conference on management of data* (pp. 205–216). ACM Press.

Lenzerini, M. (2002). Data integration: a theoretical perspective. In *Proceedings of the 21st ACM SIGMOD-SIGACT-SIGART symposium on principles of database systems (PODS-02)* (pp. 233–246). ACM Press.

Lloyd, J. W. (1987). *Foundations of logic programming*. Heidelberg, RG: Springer.

Madhavan, J., Bernstein, P., Chen, K., & Halevy, A. (2005). Corpus-based schema matching. In *Proceedings of the 21st international conference on data engineering (ICDE-05)* (pp. 57–68). IEEE Computer Society.

Melnik, S., Garcia-Molina, H., & Rahm, E. (2002). 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)* (pp. 117). IEEE Computer Society.

Nottelmann, H., & Straccia, U. (2004). A probabilistic approach to schema matching. Technical Report 2004-TR-60, Istituto di Scienza e Tecnologie dell'Informazione, Consiglio Nazionale delle Ricerche, Pisa, Italy.

Nottelmann, H., & Straccia, U. (2005). sPLMap: a probabilistic approach to schema matching. In D. E. Losada, & J. M. F. Luna, (Eds.), *27th European conference on information retrieval research (ECIR 2005)*.

Nottelmann, H., & Straccia, U. (2006). A probabilistic, logic-based framework for automated Web directory alignment. In Z. Ma (Ed.), *Soft computing in ontologies and the semantic Web. Studies in fuzziness and soft computing*. Springer-Verlag.

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

Sebastiani, F. (2002). Machine learning in automated text categorization. *ACM Computing Surveys, 34*(1), 1–47.

Straccia, U., & Troncy, R. (2005). oMAP: combining classifiers for aligning automatically OWL ontologies. In *6th International conference on Web information systems engineering (WISE-05). Lecture notes in computer science* (Vol. 3806, pp. 133–147). Springer-Verlag.