Information Retrieval and Machine Learning for Probabilistic Schema Matching

Henrik Nottelmann
University of Duisburg-Essen
Lotharstrasse 65
47048 Duisburg, Germany
nottelmann@uni-duisburg.de

Umberto Straccia
ISTI-CNR
Via G. Moruzzi 1
56124 Pisa, Italy
straccia@isti.cnr.it

ABSTRACT

Schema matching is the problem of finding correspondences (mapping rules, e.g. logical formulae) between heterogeneous schemas. This paper presents a probabilistic framework, called sPLMap, for automatically learning schema mapping rules. Similar to LSD, different techniques, mostly from the IR field, are combined. Our approach, however, is also able to give a probabilistic interpretation of the prediction weights of the candidates, and to select the rule set with highest matching probability.

Categories and Subject Descriptors: H.2.1 [Database Management]: Logical Design - Schema and subschema

General Terms: Theory, Experimentation

Keywords: sPLMap, schema matching, predicate logics, probability theory

1. INTRODUCTION

Combining heterogeneous data from different sources, i.e. transforming data structured under one schema into data structured under a different schema, is an old but emerging problem in the “information age” (e.g. for federations of existing digital libraries like the ACM DL or CiteSeer). This problem is currently under investigation in the context of information integration [3] and data exchange [2]. When schemas change frequently over time, or a large amount of schemas is involved, automatic mapping generation (called schema matching) is mandatory.

This paper describes the sPLMap framework (probabilistic, logic-based mapping between schemas) [4, 5] for automatic schema matching, which is founded on information retrieval and machine learning techniques. From well-known approaches like LSD [1], it borrows the idea of combining several specialized components (called “classifiers”) for finding the best mapping, e.g. based on attribute name comparison 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, and the usage of probability theory for estimating the degree of correctness of a learned rule. Thus, it provides a sound theoretical justification for its (optimum) selection, and a measure for the quality of a learned schema mapping.

In the following, a schema R = (R1, . . . , Rn) is a tuple of binary relations Ri containing document identifiers and attribute values (belonging to a data type, e.g. “Text”, “Name”, “Year” or “DateISO8601”). This structure corresponds to a linear list of multi-valued schema attributes. The instance of an attribute Rj is also denoted by Rj, and, similarly, R is used for the schema and the corresponding schema instance.

A schema mapping rule has the form Si → Tj. Formally, it specifies that pairs of document identifier and attribute values have to be copied from source instance Si into target instance Tj. Let Σj be a set of rules with common rule head Tj. Then, Tj = ∪k Sj denotes the result of applying all rules from Σj onto the source instance S; thus, Tj is an instance over the target attribute Tj. Then, T = (T1, . . . , Tn) denotes the instance derived by applying Σ onto S. Typically, the set Σ of mapping rules on which Tj and T depend is clear from the context and left out.

2. LEARNING SCHEMA MAPPINGS

Consider a target schema T = (T1, . . . , Tn) and a source schema S = (S1, . . . , Sn) together with two instances (which do not necessarily describe the same documents). The goal is to find the “best” set of mapping rules Σ which maximizes the probability Pr(Σ, T, S) that the tuples in T and in the mapping result T are ‘similar’, that is, given a random tuple in T, estimate the probability that it is a tuple in T and vice-versa.

Each Σ can be partitioned into t sets Σ1, . . . , Σt (where any Σj contains only rules with rule head Tj). As the rules in these sets operate independently onto different target attributes, we obtain:

Pr(Σ, T, S) = \prod_{j=1}^{T} Pr(Σ_j, T, S) .

Now, Pr(Σ_j, T, S) is estimated as the probability that a tuple in \hat{T}_j is also in T_j, and vice-versa:

Pr(Σ_j, T, S) = Pr(T_j | \hat{T}_j) · Pr(\hat{T}_j | T_j) = Pr(\hat{T}_j | T_j)^2 .

Unfortunately, there are exponentially many possible sets Σj. For computational simplification, we assume that S1 and S2 are dis-
vide good precision and rather low recall, while using data types
MARC 21 attributes). Typically, the name-based classifiers pro-
tain text which equals the concatenation of strings from multiple
for the difficult LOC collection (where the DC attributes often con-
the combination of all classifiers, while the performance is worse
cannot report here due to space limitations) satisfactory results for
well always depends on the concrete schemas (and their instances).

<table>
<thead>
<tr>
<th>Schema-based</th>
<th>Content-based</th>
</tr>
</thead>
<tbody>
<tr>
<td>CL name</td>
<td>CLL value overlap</td>
</tr>
<tr>
<td>CL stem</td>
<td>CLNN kNN</td>
</tr>
<tr>
<td>CL type</td>
<td>Naive Bayes</td>
</tr>
<tr>
<td>CLKL KL</td>
<td></td>
</tr>
</tbody>
</table>

Table 1: Classifiers

joint for $i_1 \neq i_2$. This leads us to the approximation:

$$Pr(T_i|S_T) \approx \sum Pr(S_i|T_i) .$$

Thus, in order to compute $Pr(\Sigma_i, T, S)$, the main task is to compute the $O(s \cdot t)$ probabilities $Pr(S_i|T_i)$. Similar to LSD [1], we combine different classifiers $CL_1, \ldots, CL_s$ (see table 1 for classifiers we use). Each classifier outputs a conditional probability $Pr(S_i|T_i, CL_k)$ as an approximation of $Pr(S_i|T_i)$. The final probability $Pr(S_i|T_i)$ is computed by combining the classifier predictions $Pr(S_i|T_i, CL_k)$ in a weighted sum:

$$Pr(S_i|T_i) \approx \sum_{k=1} Pr(S_i|T_i, CL_k) \cdot Pr(CL_k) .$$

In a first step, each classifier $CL_k$ computes an initial weight $w(S_i|T_i, CL_k)$, which is then normalized (due to heterogeneous scales) into the probability $Pr(S_i|T_i, CL_k) = f(w(S_i|T_i, CL_k))$. In a first normalization step, an arbitrary function can be used, e.g. linear or logistic functions. A second normalization ensures that the final values for $Pr(S_i|T_i, CL_k)$ and for $Pr(T_i|S_i, CL_k)$ are in $[0,1]$. Finally, the probability $Pr(CL_k)$ describes the probability that we rely on the judgment of classifier $CL_k$. We simply set $Pr(CL_k) = \frac{1}{s}$ (i.e., we average over all classifiers). Alternatively, one could aim at learning these probabilities for each classifiers individually.

3. EVALUATION

This section briefly describes some experimental results; more results can be found in [4, 5]. We use a 2,382 document BibTeX collection (BIBDB) with a large overlap in attribute names, and an Open Archive collection of the Library of Congress (LOC) with 1,337 entries, available in MARC 21 and Dublin Core. Both collections are split randomly into three sub-collections of approximately the same size (two for learning, one for evaluation). Classifiers from table 1 are used in isolation as well as in combination. As all LOC attributes use the same data type “Text”, and do not share common names, only the content-oriented classifiers are used for this collection.

The results in table 2 use the typical IR measures precision, recall and F-measure. For BIBDB, the combination of all classifiers yields 82% of precision and recall. According to the F-measure, the best schema classifier are $CL_{KL}$ and $CL_L$, and the worst is $CL_D$, while the best content classifier is $CL_{KL}$ and the worst is $CL_{KL}$. For LOC, the combination of all classifiers yields 64% of precision and 22% of recall only. The best content classifier is $CL_{NN}$ (slightly outperforms the combination) and the worst is $CL_{KL}$.

<table>
<thead>
<tr>
<th>Table 2: Effectiveness evaluation</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a) BIBDB</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>$CL_{NN}$</td>
</tr>
<tr>
<td>$CL_L$</td>
</tr>
<tr>
<td>$CL_{KL}$</td>
</tr>
<tr>
<td>all content</td>
</tr>
<tr>
<td>(b) LOC</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>$CL_{NN}$</td>
</tr>
<tr>
<td>$CL_{KL}$</td>
</tr>
<tr>
<td>all content</td>
</tr>
</tbody>
</table>

In our approach is that it combines neatly machine learning, information retrieval and heuristic techniques for learning a set of mapping rules. The framework has already been extended towards uncertain rules [4, 5]. In future, we plan to develop and evaluate additional classifiers $CL_k$, and learn the probabilities $Pr(CL_k)$. This framework can easily be extended towards more complex rules, which e.g. combine the content of several attributes, and to new application areas like ontologies.

4. CONCLUSION AND OUTLOOK

We have presented sPLMap, a Probabilistic, Logic-based formal framework for the important schema matching problem. The pecu-
arity of our approach is that it combines neatly machine learning, information retrieval and heuristic techniques for learning a set of mapping rules. The framework has already been extended towards uncertain rules [4, 5]. In future, we plan to develop and evaluate additional classifiers $CL_k$, and learn the probabilities $Pr(CL_k)$. This framework can easily be extended towards more complex rules, which e.g. combine the content of several attributes, and to new application areas like ontologies.

5. REFERENCES