Abstract. Schema matching is a fundamental issue to many database applications, such as query mediation and data warehousing. It becomes a challenge when different vocabularies are used to refer to the same real-world concepts. In this context, a convenient approach, sometimes called extensional, instance-based or semantic, is to detect how the same real-world objects are represented in different databases and to use the information thus obtained to match the schemas. Additionally, we argue that automatic approaches of schema matching should store provenance data about matchings. This paper describes an instance-based schema matching technique for an OWL dialect and proposes a data model for storing provenance data. The matching technique is based on similarity functions and is backed up by experimental results with real data downloaded from data sources found on the Web.

Keywords: schema matching, OWL, similarity, provenance.

1 Introduction

A database conceptual schema, or simply a schema, is a high level description of how database concepts are organized. A schema matching from a source schema \( S \) into a target schema \( T \) defines concepts in \( T \) in terms of the concepts in \( S \).

The problem of finding schema matchings becomes a challenge when different vocabularies are used to refer to the same real-world concepts (Casanova et al., 2007). In this case, a convenient approach, sometimes called extensional, instance-based or semantic, is to detect how the same real-world objects are represented in different databases and to use the information thus obtained to match the schemas. This approach is grounded on the interpretation, traditionally accepted, that “terms have the same extension when true of the same things” (Quine, 1968).

Moreover, in many applications, the schema matchings alone are not sufficient, it is also required to unveil the evidences, and to reveal the methods used to get to the final alignments. We refer to such data as the provenance data about matchings.

In this paper we address the problem of matching two schemas that belong to an expressive OWL dialect. We adopt an instance-based approach, assuming that a set of instances from each schema is available.

The major contributions of this paper are four-fold. First, we decompose the problem of OWL schema matching into the problem of vocabulary matching and the problem of concept mapping. We also introduce sufficient conditions guaranteeing that a vocabulary matching induces correct concept mappings. Second, we describe an OWL schema matching technique based on the notion of similarity. Third, we evaluate the precision of the proposed technique using data available on the Web. Finally, we propose a data model to store provenance data.


Bilke and Naumann (2005) describe an instance-based technique that explores similarity algorithms. Brauner et al. (2007a) adopt the same idea to match two thesauri. Wang et al. (2004) describe a technique to match Web databases, which uses a set of typical instances. Brauner et al. (2007b) apply this idea to match geographical database. Brauner et al. (2008) describe a matching algorithm based on measuring the similarity between attribute domains.

Unlike any of the above techniques, the schema matching process we propose uses similarity functions to induce vocabulary matchings in a non-trivial way, using an expressive OWL dialect. Through a set of examples, we also illustrate that the structure of OWL schemas may lead to incorrect concept mappings, and indicate how to avoid such pitfalls.
This paper is organized as follows. Section 2 introduces the OWL dialect adopted and the notions of vocabulary matching and concept mapping. Section 3 describes our technique to obtain OWL vocabulary mappings and contains experimental results. Section 4 describes how to induce concept mappings from vocabulary matchings. Section 5 presents a provenance data model for schema matchings. Finally, section 0 lists the conclusions and directions for future work.

2 OWL Schema Matching

2.1 OWL Extralite

We assume that the reader is familiar with basic XML concepts. In particular, recall that a resource is anything identified by an URIref and that an XML namespace or a vocabulary is a set of URIrefs. A literal is a character string that represents an XML Schema datatype value. We refer the reader to (Bechhofer et al., 2004) for more details.

An RDF statement (or simply a statement) is a triple (s,p,o), where s is a URIref, called the subject of the statement, p is a URIref, called the property of the statement, and o is either a URIref or a literal, called the object of the statement; if o is a literal, then o is also called the value of property the p.

The Web Ontology Language (OWL) describes classes and properties in a way that facilitates machine interpretation of Web content. The description of OWL is organized as three dialects: OWL Lite, OWL DL and OWL Full.

We will define and work an OWL dialect, that we call OWL Extralite. It supports:
- named classes
- datatype and object properties
- subclasses
- individuals
- domain and range of datatype and object properties
  - the domain is always a class
  - the range of a datatype property is an XML schema type, whereas the range of an object property is a class
  - minCardinality and maxCardinality, with the usual meaning
  - inverseFunctionalProperty, which captures simple keys (we note that only OWL Full supports the inverseFunctionalProperty for datatype properties)

Note that, OWL Extralite thus defined has the same expressiveness as UML (OMG, 2009). In the context of this paper we use OWL Extralite to express database schemas since it is a convenient technology to exchange data in the Web, as well as to manipulate database schemas.

An OWL schema (more often called an OWL ontology) is a collection of RDF triples that uses the OWL vocabulary. A concept in an OWL schema is a class, datatype property or object property defined in the schema. The vocabulary of the schema is the set of concepts defined in the schema (a set of URIrefs). It is important to note that, unlike UML, the scope of a property name is global to the OWL Extralite schema.

A triple of the form (s,rdf:type,c) indicates that s is an instance of a class c; a triple of the form (s,p,v) indicates that s has a datatype property p with value v; and a triple of the form (s,p,o) indicates that s and o are related by an object property p.

In the rest of the paper, we refer to OWL Extralite schemas simply as schema. Figure 1 and Figure 2 show schemas for fragments of the Amazon and the eBay databases, using a shorthand notation to save space and improve readability. Consistently with XML usage, from this point on, we will use the namespace prefixes am: and eb: to refer to the vocabularies of the Amazon and the eBay schemas respectively, and qualified names of the form V:T to indicate that T is a term of the vocabulary V.

In Figure 1, for example, am:title is defined as a datatype property with domain am:Product and range string (an XML Schema data type), am:Book is declared as a subclass of am:Product, and am:publisher is defined as an object property with domain am:Book and range am:Publ. Note that the scope of am:title and am:publisher is the schema, and not the classes defined as their domains.
Figure 1. An OWL schema for a fragment of the Amazon Database.

Figure 2. An OWL schema for a fragment of the eBay Database.

Furthermore, although not indicated in Figure 1, we assume that all properties, except `am:author`, have `maxCardinality` equal to 1, and that `am:isbn` is inverse functional. This means that all properties are single-valued, except `am:author`, which is multi-valued, and that `am:isbn` is a key of `am:Book`. Likewise, although not shown in Figure 2, all properties, except `eb:author`, have `maxCardinality` equal to 1, and `eb:isbn-10` and `eb:isbn-13` are inverse functional.

### 2.2 Vocabulary Matching and Concept Mapping

We decompose the problem of schema matching into the problem of vocabulary matching and the problem of concept mapping. In this section, we introduce both notions with the help of examples.

In what follows, let $S$ and $T$ be two schemas, and $V_S$ and $V_T$ be their vocabularies, respectively. Let $C_S$ and $C_T$ be the sets of classes and $P_S$ and $P_T$ be the sets of datatype or object properties in $V_S$ and $V_T$, respectively.

A contextualized vocabulary matching between $S$ and $T$ is a finite set $\mu$ of quadruples $(v_i, e_i, v_j, e_j)$ such that

- if $(v_i, v_j) \in C_S \times C_T$, then $e_i$ and $e_j$ are the top class $T$
- if $(v_i, v_j) \in P_S \times P_T$, then $e_i$ and $e_j$ are classes in $C_S$ and $C_T$ that must be subclasses of the domains, or the domains themselves, of properties $v_i$ and $v_j$, respectively.

If $(v_i, e_i, v_j, e_j) \in \mu$, we say that $\mu$ matches $v_i$ with $v_j$ in the context of $e_i$ and $e_j$, that $e_i$ is the context of $v_i$ and that $(e_i, v_i)$ is a contextualized concept, for $i=1,2$. A contextualized property (or class) matching is a matching defined only for properties (or classes).

Intuitively, a vocabulary matching expresses equivalences between properties and classes in a given context. The context of a property $P$ in a vocabulary matching is an RDF class that specifies the `rdf:type` of subjects of existing triples of the form `(?subject P ?object)` for which the matchings holds. The context of a class is always the top class $T$ (i.e., this notion is not used for class matchings). Note that, if the database instances follows the schema the class of the ?subject must be either
We can directly infer, for example, imagine the triples of the form $\text{am:Book} \leftarrow ?b \text{am:title} ?t$. these two databases, i.e. the Amazon data with the vocabularies of Amazon and eBay databases.

For example, Table 1 shows a fragment of the vocabulary matching between the schemas in Figure 1 and Figure 2. The first row indicates that the classes $\text{am:Book}$ and $\text{eb:Book}$ are equivalent. The last row indicates that the property $\text{am:Publ}$ applied to instances of type $\text{am:Publ}$ is equivalent to the property $\text{eb:publisher}$ applied to instances of type $\text{eb:Book}$.

<table>
<thead>
<tr>
<th>Amazon</th>
<th>eBay</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{am:Book}$</td>
<td>$\text{eb:Book}$</td>
</tr>
<tr>
<td>$\text{am:title}$</td>
<td>$\text{eb:title}$</td>
</tr>
<tr>
<td>$\text{am:author}$</td>
<td>$\text{eb:author}$</td>
</tr>
<tr>
<td>$\text{am:listPrice}$</td>
<td>$\text{eb:startPrice}$</td>
</tr>
<tr>
<td>$\text{am:name}$</td>
<td>$\text{am:Publ}$</td>
</tr>
</tbody>
</table>

A concept mapping from a source schema $S$ to a target schema $T$ is a set of transformation rules that express concepts of the target schema $T$ in terms of concepts of the source $S$ such that it is possible to translate queries over $T$ into queries over $S$.

In this paper we consider queries defined in the SPARQL Query Language for RDF (Prud’hommeaux and Seaborne, 2008). The query of Figure 3a returns titles, authors and publishers of book instances from the Amazon database. The variable $?b$ in lines 4-6 means that only instances which have the properties $\text{author}$, $\text{title}$ and $\text{publisher}$ attached to them match the WHERE criteria. The variable $?p$ in lines 6 and 7 means that, in addition to the previous criteria, only instances which are related to another instance through the property $\text{am:publisher}$ match the WHERE criteria. This is the JOIN relational operator for RDF graphs. Figure 3b shows an equivalent query over the eBay database.

Let $A$ and $E$ be the schemas of Amazon and eBay databases respectively, and $A_S$ and $E_S$ be states of these two databases, i.e. $A_S$ and $E_S$ contain individuals and their property values of the two databases. The query depicted in Figure 3a is valid over the RDF graph $A \cup A_S$.

Now consider the graph $G = E \cup A \cup A_S$, i.e. the Amazon data with the vocabularies of Amazon and eBay. Let’s see how to add triples to $G$ in order to get the same answer while submitting the previous two queries to $G$. To do that, we adopt the Semantic Web Rule Language (SWRL) (Horrocks et al., 2004), in a simpler syntax, to infer additional triples. An example of the rules in our simplified syntax would be:

1. $\text{eb:title}(b,t) \leftarrow \text{am:title}(b,t), \text{am:Book}(b)$
2. $\text{eb:Book}(b) \leftarrow \text{am:title}(b,t), \text{am:Book}(b)$

The first rule says that, if $b$ is an individual attached to the property $\text{am:title}$ and it is of class $\text{am:Book}$ then $b$ is attached to the property $\text{eb:title}$. The second rule means that if the same conditions hold then $b$ is of class $\text{eb:Book}$. Note that these two rules can be derived from the second matching depicted on Table 1 because the matching says that the property $\text{am:title}$ while attached to instances of class $\text{am:Book}$ is equivalent to $\text{eb:title}$ when subjects are of class $\text{eb:Book}$. For example, imagine the triples of the form $(b, \text{am:title}, t)$ and $(b, \text{rdf:type}, \text{am:Book})$. We can directly infer the triples $(b, \text{eb:title}, t)$ and $(b, \text{rdf:type}, \text{eb:Book})$.

We can extend the previous set of set of rules using other rows of Table 1 as follows.
3. \( eb:author(b,a) \leftarrow am:author(b,a), am:Book(b) \)
4. \( eb:Book(b) \leftarrow am:author(b,a), am:Book(b) \)
5. \( eb:startPrice(b,pr) \leftarrow am:listPrice(b,pr), am:Book(b) \)
6. \( eb:Book(b) \leftarrow am:listPrice(b,pr), am:Book(b) \)
7. \( eb:publisher(b,n) \leftarrow am:publisher(b,p), am:name(p,n), am:Book(b) \)
8. \( eb:Book(b,n) \leftarrow am:publisher(b,p), am:name(p,n), am:Publ(p) \)
9. \( eb:Book(b) \leftarrow am:Book(b) \)

Note that rule 8 is not directly derived from Table 1. We will later specify, how to derive such a rule.

Now let \( R \) be the set of 9 rules derived from vocabulary matching of Table 1 and \( R(G) \) be the set of inferred triples from \( G \) by \( R \). Then, the queries of Figure 3 over \( G \cup R(G) \) return the same set of answers.

The rules can be used to do query translation. Consider that a query over eBay should be translated into a query over Amazon. According to rule 3, triples of the form \( (b, eb:author, a) \) can be derived from triples \( (b, am:author, a), (b, rdf:type, am:Book) \). By the same rule, the triple pattern \( {?b eb:author ?author} \) can be replaced by \( {?b am:author ?author. ?b am:author ?a} \). Using rules 1, 3 and 7, the query over eBay can be translated into a query over Amazon as in Figure 4. The translated query can be simplified if we consider that the domain of \( am:publisher \) is \( am:Book \) and the domain of \( am:name \) is \( am:Publ \). In this case lines 5 and 9 can be omitted.

### Figure 4. Translate SPARQL query from Figure 2

```
1. PREFIX am:<...>
2. SELECT ?title ?author ?pub
3. WHERE
4.   {?b am:author ?author.
5.    ?b rdf:type am:Book.
9.    ?p rdf:type am:Publ}
```

3. **Instance-based Vocabulary Matching**

#### 3.1 Instance-based Technique

In this section, we describe an instance-based process to create contextualized vocabulary matchings that are structurally consistent.

Let \( S \) and \( T \) be two (OWL Extralite) schemas, and \( V_S \) and \( V_T \) be their vocabularies, respectively. Let \( C_S \) and \( C_T \) be the sets of classes, and \( P_S \) and \( P_T \) be the sets of datatype or object properties in \( V_S \) and \( V_T \), respectively.

A *contextualized vocabulary matching* between \( S \) and \( T \) is a finite set \( \mu_V \) of quadruples \( (v_i,e_i,v_j,e_j) \) such that

- (i) if \( (v_i,v_j) \in C_S \times C_T \) then \( e_i \) and \( e_j \) are the top class \( T \)
- (ii) if \( (v_i,v_j) \in P_S \times P_T \) then \( e_i \) and \( e_j \) are classes in \( C_S \) and \( C_T \) that must be subclasses of the domains of \( v_i \) and \( v_j \), respectively
- (iii) and these are the only possible quadruples in \( \mu_V \)

If \( (v_i,e_i,v_j,e_j) \in \mu_V \), we say that \( \mu_V \) matches \( v_i \) with \( v_j \) in the context of \( e_i \) and \( e_j \), that \( e_i \) is the context of \( v_i \) and that \( (v_i,e_i) \) is a contextualized concept, for \( i=1,2 \). A contextualized property (or class) matching is a matching defined only for properties (or classes).
We first recall the matching technique for catalogue schemas based on similarity heuristics introduced in (Leme et al., 2008a). Briefly, a catalogue is a relational database whose schema $S$ has a single table. Given a catalogue state $U_S$, an attribute $A$ of $S$ is represented by the set of values of $A$ that occur in $U_S$, or by the set of pairs $(i,v)$ such that $v$ is the value of $A$ for the object with id $i$ that occurs in $U_S$. If the domain of $A$ is a set of strings, the set of values is replaced by a set of tokens, and the attribute representations are reinterpreted accordingly. Similarity models were then applied to such attribute representations to generate attribute matchings between two catalogue schemas.

Bilke and Naumann (2005) propose an instance matching technique where each database tuple is represented by a character string, created by concatenating all attribute values of each tuple. The technique uses k-mean clustering algorithms to find duplicate tuples. The identification of duplicates is necessary for creating $(i,v)$ representations of attributes. However, we note that the representations of the same object in distinct databases may differ in the list of attributes and/or in the attribute values. As a consequence, we may end up with dissimilar tuples that are used to represent the same object.

For example, suppose that we apply the Bilke and Naumann technique to match the two instances that represent the book “The Tragedy of Romeo and Juliet”, whose property-value pairs are shown in Table 2. If we measure the similarity between the sets of tokens by the percentage of common tokens extracted from all property values of each instance, we obtain a score of 43% of common tokens. By contrast, if we consider only the values of properties that match, the similarity increases to 70%. Please note that, to improve the instance matching strategy, we used the fact that am:Book matches eb:Book, and the fact that several other properties match.

Combining these observations, we propose the four-step vocabulary matching process outlined as follows:

1. Generate a preliminary property matching using similarity functions.
2. Use the property matching obtained in Step (1) to generate a class matching.
3. Use the property matching obtained in Step (1) to generate an instance matching.
4. Use the class matching and the instance matching obtained in Step (2) to generate a refined contextualized property matching.

The final vocabulary matching is the result of the union of the class matching obtained in Step (2) and the refined property matching obtained in Step (4). Step (1) generates a preliminary property matching based on the intuition that “two properties match iff they have many values in common and few values not in common”. Step (2) creates class matchings that reflect the intuition that “two classes match iff they have many matching properties”. To work correctly, Step (2) and (3) require that Step (1) generates preliminary property matchings that do not use $(i,v)$ pairs to represent properties.

In what follows, let $S$ and $T$ be two schemas, $V_S$ and $V_T$ be their vocabularies, $P_S$ and $P_T$ be their sets of properties, and $C_S$ and $C_T$ be their sets of classes, respectively. Let $U_S$ and $U_T$ be fixed sets of triples of $S$ and $T$, respectively, to be used to compute the vocabulary matchings.

**Table 2.** Example the same book instance representation in eBay and Amazon.

<table>
<thead>
<tr>
<th>eBay</th>
<th>Amazon</th>
</tr>
</thead>
<tbody>
<tr>
<td>isbn-10 = “039577537X”</td>
<td>isbn = “039577537X”</td>
</tr>
<tr>
<td>isbn-13 = 9780395775370</td>
<td>ean = 9780395775370</td>
</tr>
<tr>
<td>title = “The Tragedy of Romeo and Juliet”</td>
<td>title = “Tragedy of Romeo and Juliet: And Related Readings (Literature Connections)”</td>
</tr>
<tr>
<td>author = “William Shakespeare”</td>
<td>author = “William Shakespeare”</td>
</tr>
<tr>
<td>publisher = “Houghton Mifflin”</td>
<td>name = “Houghton Mifflin Company”</td>
</tr>
<tr>
<td>returnPolicyDetails = “NO RETURNS ARE ACCEPTED”</td>
<td>-</td>
</tr>
<tr>
<td>condition = “Like New”</td>
<td>-</td>
</tr>
<tr>
<td>binding = “Hardcover”</td>
<td>-</td>
</tr>
<tr>
<td>listPrice = 18.92</td>
<td>currency = “USD”</td>
</tr>
</tbody>
</table>
Step (1): Preliminary property matching

Let $\mathcal{U}$ be the universe of all tokens extracted from literals and all URIs. Consider a similarity function $\sigma: \mathcal{U} \times \mathcal{U} \rightarrow [0,1]$, a similarity threshold $\tau \in [0,1]$ and a related similarity threshold $\tau' \in [0,1]$ such that $\tau' < \tau$.

For each property $P \in P_s$, for each class $C \in C_s$ such that $C$ is the domain of $P$ or a subclass of the domain of $P$, consider the contextualized property $P^C = (P,C)$ and construct the set of observed-value representations of all $v$ such that either there are triples of the form $(I,P,v)$ and $(I,rdf:type,C')$ in $U_S$ or there are triples of the form $(I,P,s)$ and $(I,rdf:type,C')$ in $U_S$ where $v$ is a token in the literal string $s$, where $C' = C$ or $C'$ is a subclass of $C$, and likewise for a property in $P_T$. We call $o[U_S,P^C]$ the observed-value representation of $P^C$ in $U_S$. This construction explores the fact that $P$ is inherited by all subclasses of its domain.

The contextualized property matching $\mu_P$ between $S$ and $T$ induced by $\sigma$ and $\tau$ and based on the observed-value representation of properties, is the relation $\mu_P$ such that

$$(P,C,Q,D) \in \mu_P \text{ iff } \sigma(o[U_S,P^C],o[U_T,Q^D]) \geq \tau$$

(1)

Step (2): Class matching

For each class $C$ in $C_s$, let $\text{props}[S,C]$ be the set of properties in $P_S$ whose domain is $C$ or that $C$ inherits from its superclasses, and likewise for classes in $C_T$. We call $\text{props}[S,C]$ the property representation of $C$ in $U_S$.

The contextualized class matching $\mu_C$ between $S$ and $T$ induced by $\sigma$, $\tau$ and $\mu_P$ is the relation $\mu_C \subseteq C_S \times C_T$ such that (recall that $T$ is the top class)

$$(C,T,D,T) \in \mu_C \text{ iff } \sigma(\text{props}[S,C],\text{props}[S,T,D])) \geq \tau$$

(2)

where $\text{props}[S,C,T,D] = \text{relprops}[S,C,T,D] \cup \text{not}_\text{relprops}[S,C,T,D]$, $\text{relprops}[S,C,T,D]$ denotes the set of properties $P$ of class $C$ of $S$ such that there is a property $Q$ of the class $D$ of $T$ such that $(P,C,Q,D) \in \mu_P$, and where $\text{not}_\text{relprops}[S,C,T,D]$ denotes the set of properties $P$ of the class $D$ of $T$ such that there are not related properties in $S$ by $\mu_P$. Note that it does not make sense to directly compute $\sigma(\text{props}[S,C],\text{props}[T,D])$, since $\text{props}[S,C]$ and $\text{props}[T,D]$ are sets of URIs from different vocabularies. To avoid this problem, we replaced $\text{props}[T,D]$ by $\text{props}[S,C,T,D]$.

Step (3): Instance matching

From the matchings directly induced by $\sigma$ and $\tau$, computed in the previous step, the process then derives an instance matching and a refined contextualized property matching, as follows.

Figure 5 shows the algorithm that computes the instance matching. Its inputs are $S$, $T$, class matching $\mu_C$ and property matching $\mu_P$. It also implicitly receives $U_S$ and $U_T$ as input. It outputs the instance matching $\mu_I$ that relates matching class instances in $U_S$ and $U_T$.

In Figure 5, if $C$ is a class in $C_S$ and $J$ is an instance of $C$ in $U_S$, then $t_I[S,C,I,T,D]$ denotes the set of tokens extracted from all values $v$ such that, for some property $P \in P_S$, for some property $Q$ in $P_T$, for some class $D \in C_T$, there is a triple $(I,P,v)$ in $U_S$ and there is a quadruple $(P,C,Q,D)$ in $\mu_P$. In addition, if $C$ is a class in $C_S$ and $J$ is an instance of $D$ in $U_T$, then $t_I[S,C,T,D]$ denotes the set of tokens extracted from all values $v$ such that, for some property $P \in P_S$, for some property $Q$ in $P_T$, for some class $D \in C_T$, there is a triple $(I,Q,v)$ in $U_T$ and there is a quadruple $(P,C,Q,D)$ in $\mu_P$.

![Figure 5. The class instance matching algorithm.](image-url)
Step (4): Refined property matching

Figure 6 shows the algorithm that computes the refined contextualized property matching. It depends on the following additional definitions. For each \((P,C,Q,D) \in \mu_P\) such that \((C,T,D) \in \mu_C\), construct the set \(q\) of triples \((I,u,v)\) such that there are triples of the form \((I,P,u)\) and \((I,\text{rdf:type},C)\) in \(U_5\), there are triples of the form \((J,Q,v)\) and \((J,\text{rdf:type},D)\) in \(U_7\), and \((I,C,J,D) \in \mu_P\) (where \(\mu_P\) is the instance matching of Figure 3). Define \(iv(P,C,Q,D) = (s,t)\) such that \(s = \{I,u\}/(\exists v)(I,u,v) \in q\) and \(t = \{I,v\}/(\exists u)(I,u,v) \in q\). We call \(s\) the instance-value representation of \(P^C\) in \(U_5\) (and likewise for \(t\)). This second representation is useful since it helps distinguish between properties with similar sets of values, that refer to distinct instances, matched by \(\mu_P\).

Returning to the algorithm in Figure 6, it has similar inputs to the algorithm depicted in Figure 5. Its output, however, is the contextualized property matching \(\mu_C\) between properties whose domains are classes directly or indirectly matched by \(\mu_C\). The algorithm uses the maximum similarity values computed using the observed-value and the instance-value representations for a pair of properties \(P\) and \(Q\), and the more relaxed similarity threshold. Although not shown in Figure 6, object properties receive a special treatment, since their representations are sets of URIs that are compared with help of the instance matching \(\mu_C\) (computed by the algorithm in Figure 5).

\[
\text{CONTEXTUALIZED-PROPERTY-MATCHING}(S,T,\mu_C,\mu_I)
\]

for each pair of classes \((C,D)\) in \(S\) and \(T\)

such that \(\mu_C\) matches \(C\) with \(D\)

or \(C'\) dominates \(C\) and \(\mu_C\) matches \(C'\) with \(D\)

or \(\mu_I\) matches \(C\) with \(D'\) and \(D'\) dominates \(D\)

for each pair \((P,Q)\) of properties of \(C\) and \(D\)

\[
X = \sigma(\sigma([U_6, P^C], \sigma([U_7, Q^C])))
\]

if \((C\) matches \(D)\) then

\[
(s,t) = iv(P,C,Q,D)
\]

\[
Y = \sigma(s,t)
\]

else

\[
Y = 0
\]

if \(\max(X,Y) \leq \tau\) then

\[
\mu_C = \mu_C \cup (P,C,Q,D)
\]

Figure 6. The contextualized property matching algorithm.

The final vocabulary matching \(\mu\) is the union of the class matching \(\mu_C\) induced by \(\sigma\), \(\tau\) and \(\mu_P\) and the contextualized property matching \(\mu_I\) computed by the algorithm in Figure 6.

3.2 Experimental Vocabulary Matching Results

We conducted an experiment to assess the performance of the vocabulary matching process described in section 3.1, using product data obtained from Amazon and eBay websites.

We preferred to use data directly downloaded from the Web, rather than using the benchmark proposed in (Duchateau et al., 2007), because the last does not include instances and, therefore, is unsuitable to test the proposed process.

We first defined a set of terms, which was used to query both databases. From the query results, we extracted the less frequent terms common in both databases. We then used this set of terms to query the databases once more. This pre-processing step enhanced the probability of retrieving duplicate objects from the databases, which is essential to evaluate any instance-based schema matching technique. We extracted a total of 116,201 records: 16,410 from Amazon and 99,791 from eBay.

As similarity functions we adopted the contrast model (Leme et al., 2008a) for property matchings, and the cosine distance with TF/IDF for instance matchings. The experiments provided us with enough empirical data to conclude that the contrast model performs better in situations where the goal is to emphasize the differences between two sets of values. This follows because the contrast model allows for parameter calibration.

Table 3 shows sample entries of the vocabulary matching obtained. The headings indicate that \(e_1\) is the context of \(v_1\), and \(e_2\) that of \(v_2\). Also, “B” abbreviates classes \(\text{eb:Book}\) and \(\text{am:Book}\).
The rightmost column of Table 3 classifies the matchings in types: \( tp \) for true positive, \( fp \) for false positive and \( fn \) for false negative. Since the total number (not all shown in Table 3) of true positives is 25, that of false positives is 4 and that of false negatives is 10, the performance measures therefore are:

\[
\text{precision} = \frac{tp}{tp + fp} = 86\% \quad \text{recall} = \frac{tp}{tp + fn} = 71\% \quad F = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} = 78\%
\]

Lines 3, 5 and 6 of Table 3 refer to matchings that would have been considered false negatives, if the algorithm in Figure 4 ignored the instance-value representation of properties. In this case, the performance measures would drop to:

\[
\text{precision} = 82\% \quad \text{recall} = 51\% \quad f\text{Measure} = 63\%
\]

<table>
<thead>
<tr>
<th>#</th>
<th>eBay</th>
<th>Amazon</th>
<th>Match Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>v1</td>
<td>Books</td>
<td>Books</td>
<td>tp</td>
</tr>
<tr>
<td>v2</td>
<td>author</td>
<td>author</td>
<td>tp</td>
</tr>
<tr>
<td>4</td>
<td>format</td>
<td>binding</td>
<td>tp</td>
</tr>
<tr>
<td>5</td>
<td>isbn-10</td>
<td>isbn</td>
<td>tp</td>
</tr>
<tr>
<td>6</td>
<td>editionDesc</td>
<td>format</td>
<td>fp</td>
</tr>
<tr>
<td>7</td>
<td>Offer</td>
<td>Books</td>
<td>fp</td>
</tr>
</tbody>
</table>

### 4 Concept Mapping Induced by Vocabulary Matching

#### 4.1 Definition

Let \( S \) be an OWL Extralite schema in what follows.

We say that \( S \) is well-formed iff

- for any property \( p \) of \( S \), the domain of \( p \) is a class of \( S \)
- for any object property \( p \) of \( S \), the range of \( p \) is a class of \( S \)
- for any class \( c \) of \( S \), if \( s \) is defined as a superclass of \( c \) in \( S \), then \( s \) is also a class of \( S \)

We understand \( S \) as a theory \( T[S]=(A[S],C[S]) \) in \( \mathcal{ALCQI} \) (Chomicki and Saake, 1998), a dialect of Description Logics, such that

- the concepts and roles of the alphabet \( A[S] \) are the classes and properties of \( S \)
- the axioms of \( C[S] \) are the constraints of \( S \), denoted in \( \mathcal{ALCQI} \) as follows:
  - a property \( p \) has domain \( d \) and range \( r \): \( T \models \forall p.r \land \forall p.d \)
  - a property \( p \), with range \( r \), is inverse functional: \( r \models (\leq 1 p^-) \)
  - a property \( p \), with domain \( d \), has minCardinality \( k \): \( d \models (\geq k p) \)
  - a property \( p \), with domain \( d \), has maxCardinality \( k \): \( d \models (\leq k p) \)
  - a class \( s \) is defined as a superclass of \( c \): \( c \models s \)

In what follows, we will also use from \( \mathcal{ALCQI} \) the intersection of two concepts, denoted \( c \cap d \), and the subsumption of two concepts, denoted \( c \sqsubseteq d \).
Let $V$ be the set of variables, which is assumed to be disjoint from the set of concepts of $S$. A class \textit{literal} is an expression of the form $c(x)$, where $c$ is a class and $x$ is a variable; a property \textit{literal} is an expression of the form $p(x,y)$, where $p$ is a property and $x$ and $y$ are variables; a \textit{literal} is a class literal or a property literal. A \textit{conjunction} is a list of literals separated by commas. A \textit{disjunction} is a list of conjunctions separated by semi-colons. (This notation should be familiar to Prolog programmers).

A \textit{rule} is an expression of one of the forms:

- $c(x) \leftarrow \neg B[x]$, where $c(x)$ is a class literal and $B[x]$ is a disjunction where the variable $x$ occurs in each conjunction.
- $p(x,y) \leftarrow \neg B[x,y]$, where $p(x,y)$ is a property literal and $B[x,y]$ is a disjunction where the variables $x$ and $y$ occur in each conjunction.

The literal is the head and the disjunction is the body of the rule. We use the notation $B[x]$ and $B[x,y]$ to stress which variables must occur in the body.

Let $I$ be a set of triples of $S$. The \textit{universe} of $I$ is the set $U[I]$ of all URIrefs and literals that occur in triples of $I$.

Consistently with the notion of interpretation of Description Logics, given a class $c$ of $S$, the \textit{interpretation of $c$ in $I$} is the set

$$c^I = \{ i \in U[I] / (i,:v\in c) \in I \}$$

and, given a property $p$ of $S$, the \textit{interpretation of $p$ in $I$} is the binary relation

$$p^I = \{ (i,o) \in U[I] \times U[I] / (i,p,o) \in I \}$$

The \textit{interpretation of the intersection of two concepts $c \cap d$} is the set

$$(c \cap d)^I = c^I \cap d^I$$

We say that the subsumption of two concepts $c \subseteq d$ is \textit{true in $I$}, denoted $I \models c \subseteq d$, iff $c^I \subseteq d^I$.

Rather than resorting to the formalization in $\mathcal{ALCQI}$, we directly define when a constraint $\sigma$ of $S$ is \textit{true in $I$}, denoted $I \models \sigma$:

- if $\sigma$ declares that a property $p$ has domain $d$ and range $r$, then $I \models \sigma$ iff $p^I \subseteq d^I \times r^I$.
- if $\sigma$ declares that a property $p$, with range $r$, is inverse functional, then $I \models \sigma$ iff, for any $b \subseteq r^I$, $|\{a \in U[I] / (a,b) \in p^I\}| \leq 1$.
- if $\sigma$ declares that a property $p$, with domain $d$, has minCardinality $k$, then $I \models \sigma$ iff, for any $a \subseteq d^I$, $|\{b \in U[I] / (a,b) \in p^I\}| \geq k$.
- if $\sigma$ declares that a property $p$, with domain $d$, has maxCardinality $k$, then $I \models \sigma$ iff, for any $a \subseteq d^I$, $|\{b \in U[I] / (a,b) \in p^I\}| \leq k$.
- if $\sigma$ declares that a class $s$ is a superclass of $c$, then $I \models \sigma$ iff $c^I \subseteq s^I$.

We now turn to the semantics of rules. A \textit{valuation} for the set of variables $V$ in $I$ is a function $v$ that maps the variables in $V$ into elements of $U[I]$.

We extend the notion of interpretation of the right-hand side of the rule as follows. We first define when the right-hand side of rule $B$ is \textit{true in $I$} for $v$, denoted $I,v \models B$, inductively as follows:

- if $B$ is of the form $c(x)$ then $I,v \models B$ iff $v(x) \in c^I$.
- if $B$ is of the form $p(x,y)$ then $I,v \models B$ iff $(v(x),v(y)) \in p^I$. 

• if $B$ is of the form $C,D$ then $I,v \vDash B$ iff $I,v \vDash C$ and $I,v \vDash D$
• if $B$ is of the form $C,D$ then $I,v \vDash B$ iff $I,v \vDash C$ or $I,v \vDash D$

The interpretation of the right-hand side of a rule of the form $B[x]$ in $I$ is the set

$$B[x]^I = \{ a \in U[I] \mid \text{there is a valuation } v \text{ for } V \text{ in } I \text{ such that } I,v \vDash B[x] \text{ and } v(x) = a \}$$

and the interpretation of the right-hand side of a rule of the form $B[x,y]$ in $I$ is the binary relation

$$B[x,y]^I = \{ (a,b) \in U[I] \times U[I] \mid \text{there is a valuation } v \text{ for } V \text{ in } I \text{ such that } I,v \vDash B[x,y] \text{ and } v(x) = a \text{ and } v(y) = b \}$$

Finally, we say that a set $I$ of triples of $S$ is consistent iff $I$ satisfies all constraints of $S$.

### 4.2 Derivation from Vocabulary Matching

Given an OWL schema, we say that a class $f$ dominates a class $c$ or the intersection $c = d \cap e$ of two classes $d$ and $e$ iff there is a sequence $(c_i, c_2, \ldots, c_n)$ such that

- $f = c_i$ and $c = c_n$
- $c_{i+2}$ subsumes $c_i$
- for each $i \in \{1, n-2\}$, either
  - $c_{i+1}$ and $c_i$ are classes and $c_{i+1}$ is declared as a subclass of $c_i$ or
  - $c_{i+1}$ is a class, $c_i$ is an object property and $c_{i+1}$ is declared as the range of $c_i$, or
  - $c_{i+1}$ is an object property, $c_i$ is a class and $c_{i+1}$ is declared as the domain of $c_{i+1}$

We also say that $\pi = (c_1, c_2, \ldots, c_n)$ is a dominance path from $c$ to $d$ and $\theta = (p_{k_1}, p_{k_2}, \ldots, p_{k_n})$, the subsequence of $\pi$ consisting of the object properties that occur in $\pi$, is the property path corresponding to $\pi$ (note that $\theta$ may be the empty sequence).

Let $S$ and $T$ be two (OWL Extralite) schemas in what follows. Recall that a contextualized vocabulary matching between $S$ and $T$ is a finite set $\mu_v$ of quadruples $(v_i, e_i, v_j, e_j)$.

A contextualized vocabulary matching $\mu_v$ from $S$ into $T$ is structurally correct iff, for all $(v_i, e_i, v_j, e_j) \in \mu_v$ such that $v_i$ and $v_j$ are properties:

(i) there is a class $f$ of $S$ such that $\mu_v$ matches $f$ with the domain of $v_j$ and $f$ dominates $d_j \cap e_j$, where $d_j$ is the domain of $v_j$
(ii) if $v_j$ is a datatype property, then the range of $v_j$ is a subtype of the range of $v_j$
(iii) if $v_i$ is an object property, then $\mu_v$ matches the range of $v_i$ with the range of $v_j$

Let $\mu_v$ be a structurally correct contextualized vocabulary matching. A concept mapping $M$ from $S$ into $T$ induced by $\mu_v$ is a set of rules derived from the quadruples of $\mu_v$ as follows.

For each quadruple $(v_i, e_i, v_j, e_j) \in \mu_v$, the concept mapping $M$ contains the following rules:

**Case 1:** $v_i$ and $v_j$ be classes. Then, $M$ contains rules of the form

$$v_i(x) \leftarrow v_j(x) \quad s(x) \leftarrow v_i(x) \quad \text{for each superclass } s \text{ of } v_j$$

**Case 2:** $v_i$ and $v_j$ are properties. Let $d_i$ and $d_j$ be the domains, and $r_i$ and $r_j$ be the ranges of $v_i$ and $v_j$ (recall that $r_i$ and $r_j$ are XML Schema data types, if $v_i$ and $v_j$ are datatype properties, and that $\mu_v$ matches the range of $v_i$ with the range of $v_j$, if $v_i$ and $v_j$ are object properties).

**Case 2.1:** $\mu_v$ matches $d_i$ with $d_j$. Then, $M$ contains a rule of the form
\(v_2(x, y) \leftarrow v_1(x, y), e_f(x)\)

**Case 2.2:** \(\mu_V\) does not match \(d_1\) with \(d_2\). Let \(f\) be a class of \(S\) such that \(\mu_V\) matches \(f\) with \(d_2\) and \(f\) dominates \(d_1 \cap e_i\). Let \(p_{k_1}, p_{k_2}, ..., p_{k_m}\) be the property path corresponding to a dominance path from \(f\) to \(d_1 \cap e_i\). Then, \(M\) contains a rule of the form

\(v_2(x, y) \leftarrow p_{k_1}(x, x_1), p_{k_2}(x_1, x_2), ..., p_{k_m}(x_{m-1}, z), v_f(z), e_j(z)\)

if the property path is nonempty; otherwise the rule reduces to that of case 2.1. (Note that, since \(\mu_V\) is structurally correct, a dominance path from \(f\) to \(d_1\) indeed exists. Also note that, since the dominance path may not be unique, the concept mapping induced by \(\mu_V\) is not unique).

Note that the contextualized vocabulary matching \(\mu_V\) may have more than one quadruple for the same concept \(v_1\) of the target schema, which implies that the above process may generate more than one rule for \(v_2\). In addition, \(v_2\) may be a superclass of more than one class which, again, implies that the process described in Case 1, may generate more than one rule for \(v_2\). Therefore, as a last step in the construction of the concept mapping \(M\), we collect all rules for \(v_2\) in a single rule with a disjunctive body. More precisely, if \(v_2\) is a class, and the above process generates rules of the following form

\(v_2(x) \leftarrow B_i[x], \text{ for } i \in [1, n]\)

then, we replace all such rules by a single rule \(\rho\) of the form

\(v_2(x) \leftarrow B_1[x] \cup ... \cup B_n[x]\)

and likewise, if \(v_2\) is a property.

We say that a rule \(\rho\) in \(M\) defines a concept \(v_2\) of \(T\) iff the head of \(\rho\) is of the form \(v_2(x)\), if \(v_2\) is a class, or of the form \(v_2(x, y)\), if \(v_2\) is a property (by the transformation described above, \(M\) has at most one rule for each concept of \(T\)).

However, there might be a concept \(v_2\) of \(T\) such that \(M\) has no rule that defines \(v_2\). We therefore define \(T/M\) as the subset of \(T\) restricted to the concepts that \(M\) defines. Then, the constraints of \(T/M\) are the constraints of \(T\) defined over such vocabulary. In particular, we can prove that superclasses, domains and ranges are properly defined in \(T/M\).

**Proposition 1:** Let \(\mu_V\) be a structurally correct contextualized vocabulary matching and \(M\) be a concept mapping from \(S\) into \(T\) induced by \(\mu_V\). Then:

(i) for any class \(c\) of \(T/M\), if \(s\) is a superclass of \(c\) in \(T\), then \(s\) is also a class of \(T/M\).

(ii) for any property \(p\) of \(T/M\), the domain of \(p\) is also a concept of \(T/M\).

(iii) for any object property \(p\) of \(T/M\), the range of \(p\) is also a concept of \(T/M\).

**Proof**

(i) Let \(c\) be a class of \(T/M\) and \(s\) be a superclass of \(c\) in \(T\). Since \(c\) is a class of \(T/M\), by Case 1 of the construction of \(M\), there is a rule in \(M\) of the form \(c(x) \leftarrow p_{(x)}(x)\). Since \(s\) is a superclass of \(c\), again by Case 1, there is a rule in \(M\) of the form \(s(x) \leftarrow p_{(x)}(x)\). Hence, \(s\) is defined in \(M\), that is, \(s\) is a class of \(T/M\).

(ii) Let \(p\) be a property of \(T/M\). Let \(d\) be domain of \(p\). Since \(p\) is a property of \(T/M\), by Case 2, there is a rule in \(M\) of the form \(p(x, y) \leftarrow B[x, y]\) and a class \(f\) of \(S\) such that \(\mu_V\) matches \(f\) with the domain \(d\) of \(p\). Then, by Case 1, there is a rule in \(M\) of the form \(d(x) \leftarrow f(x)\). Hence, \(d\) is defined in \(M\), that is, \(d\) is a class of \(T/M\).

(iii) Let \(p\) be an object property of \(T/M\). Let \(r\) be the range of \(p\). Since \(p\) is an object property of \(T/M\), by Case 2, there is a rule in \(M\) of the form \(p(x, y) \leftarrow B[x, y]\) and a class \(g\) of \(S\) such that \(\mu_V\) matches \(g\) with the range \(r\) of \(p\). Then, by Case 1, there is a rule in \(M\) of the form \(r(x) \leftarrow g(x)\). Hence, \(r\) is defined in \(M\), that is, \(r\) is a class of \(T/M\).

**Corollary 1:** \(T/M\) is a well-defined OWL Extralite schema.
Finally, we define the function $\overline{M}$ induced by $M$ as the mapping from sets of triples of $S$ into sets of triples of $T/M$ such that, for each set of triples $I$ of $S$, $J = \overline{M}(I)$ iff, for each rule $\rho$ in $M$

- if $\rho$ is of the form $c(x) \leftarrow B[x]$, then $J$ contains a triple $(i, : type, c)$ iff $i \in B[x]^I$
- if $\rho$ is of the form $p(x,y) \leftarrow B[x,y]$, then $J$ contains a triple $(i,p,j)$ iff $(i,j) \in B[x,y]^I$

We stress that $M$ is used to map queries submitted to the target schema $T$ into queries of the source schema $S$, whereas $\overline{M}$ is a theoretical device to prove the consistency of the concept mapping, as discussed in the next section.

4.3 Consistency

In this section we briefly discuss the consistency of OWL Extralite vocabulary matchings, referring the reader to (Leme, 2009) for the detailed definitions and proofs.

In what follows, we use the notion of subsumption as in Description Logic. We say that a class $c$ dominates a class $d$ iff there is a sequence $(c_1,c_2,\ldots,c_n)$ of classes such that $c = c_1$, $d = c_n$ and, for each $i \in [1,n-2)$, either $c_{i+1}$ is declared as a subclass of $c_i$ or there is an object property whose domain is $c_i$ and whose range is $c_{i+1}$, and $c_n$ subsumes $c_1$. We consider that a class dominates itself.

A contextualized vocabulary matching $\mu$ from $S$ into $T$ is structurally correct iff, for all $(v_1, e, v_2, e_2) \in \mu$ such that $v_1$ and $v_2$ are properties:

(i) there is a class $f$ of $S$ such that $\mu$ matches $f$ with the domain of $v_2$ and $f$ dominates $e_2$ (recall from the definition of vocabulary matching that $e_1$ is a subclass of the domain of $v_1$)
(ii) if $v_1$ is a datatype property, then the range of $v_1$ is a subtype of the range of $v_2$
(iii) if $v_1$ is an object property, then $\mu$ matches the range of $v_1$ with the range of $v_2$

A concept mapping $\gamma$ from $S$ into $T$ induced by a structurally correct contextualized vocabulary matching $\mu$ is a set of rules derived from $\mu$ as suggested by the examples in Section 2.2. The rules in $\gamma$ in turn induce a function $\overline{\gamma}$ that maps sets of triples of $S$ into sets of triples of $T$.

We say that the declarations of the domain and range of properties, property characteristics, cardinality restrictions, and subclass declarations are the constraints of a schema.

We denote the minCardinality and the maxCardinality of a property $p$ by $mC[p]$ and $MC[p]$, respectively. By convention, we take $mC[p] = 0$ (and $MC[p] = \infty$), if minCardinality (or maxCardinality) is not declared for $p$.

A property $q$ is no less constrained than a property $p$ iff $mC[p] \leq mC[q]$ and $MC[p] \geq MC[q]$ and, if $p$ is declared as inverse functional, then so is $q$. Note that this definition applies even if $p$ and $q$ are from different schemas.

Let $S$ and $T$ be two schemas, $\mu$ be a structurally correct contextualized vocabulary mapping from $S$ into $T$, and $\gamma$ be a concept mapping from $S$ into $T$ induced by $\mu$.

Let $\rho$ be a rule in $\gamma$ of the form $p(x,y) \leftarrow B[x,y]$. By construction, $p$ is a property of $T$ and all classes and properties that occur in $B[x,y]$ belong to $S$. We introduce a property of $S$, denoted $prop[B]$, defined by $B[x,y]$. We say that $\rho$ is correct iff $prop[B]$ is no less constrained than $p$. We then say that $\gamma$ is correct iff all rules in $\gamma$ are correct.

Finally, we say that a constraint $\alpha$ of $T$ is relevant for $\gamma$ iff $\alpha$ uses only concepts that occur in the heads of the rules in $\gamma$. We then say that $\gamma$ is consistent iff, if $I$ is a consistent set of triples of $S$, then the set of triples of $T$ defined by $J = \gamma(I)$ satisfies all constraints of $T$ that are relevant for $\gamma$.

**Lemma 1:** Let $\mu$ be a structurally correct contextualized vocabulary matching and $\gamma$ be a concept mapping from $S$ into $T$ induced by $\mu$. Assume that $\gamma$ is correct. Then, $\gamma$ is consistent.

(The proof generalizes Examples 2, 3 and 4. See [Leme, 2009] for the details).
5 Storing Provenance Data for Matchings

In this section we discuss the problem of storing provenance data for schema matchings and propose a data model for provenance applied to the algorithm presented in section 2.

Clearly, schema matching process is a laborious task. Automatic or semi-automatic tools that are able to identify such correspondences definitely boost up the process. Nevertheless, schema matching algorithms, in general, must be calibrated so as to achieve better performance in relation to false positives and negatives. Leme et al. (2008) propose a cross validation process which aims at choosing the best similarity model and calibrations for a given set of test data. In this context, provenance data could be used to store parameters and calibrations for each matching result in order to allow for the identification of the best models.

 Benchmarks are also very important to refine matching algorithms and to identify best suited scenarios for each algorithm. Of course the algorithms must be compared over the same dataset, otherwise it does not make sense to compare performance measures. A classification of the scenarios might also be useful. If schemas are classified according their application domains, we could identify the ones that work better in the geographic domain, for instance.

Finally, matching entries can be validated by using a semi-automated matching process. In this case, we would make the internal representation of schema elements, the intermediate matching calculations and the final similarity degree between elements available. This information would help users to decide and/or validate the matchings.

Our provenance model consists of an OWL schema (top left of Figure 7), modeled as an aggregation of elements, specialized into classes, properties, and instances. A matchable is any object that can be matched to another objet (classes or properties).

Each schema is associated with one or more datasets (bottom left of Figure 7). Each dataset contains a set of triples, which describes the elements of the schema, including instances of classes and properties. A schema also has a set of features, that can be used to identify the best model for a given scenario. For example, if feature[0] contains the classification categories of the schemas, it would be possible to select the best algorithms for the particular domain of feature[0].

From a dataset, representations (Set in Figure 7) for each matchable of the schema are extracted (Leme et al., 2009b). Each representation is a set of values, and has a type. For example, a property can be represented by a set of tokens extracted from its observed values, in this case the set is of type Token and values are the extracted tokens of the property values. Our matching technique proposes the following representation sets for each type schema elements:

(i) Classes:
   a. Set of properties (denoted by props)
(ii) Properties (datatype and object properties)
   a. Set of tokens (denoted by o)
   b. Set of instance value (i,v) pairs (denoted by iv)
(iii) Instances
   a. Set of tokens (denoted by t)

The Matcher (top right of Figure 7), stores descriptions of matching algorithms, or matchers, and of similarity functions (Similarity in Figure 7). To model the fact that a matching algorithm has a series of matching steps, as in Section 3.1, a matcher is modeled as an aggregation of matchers. Each matcher applies one or more similarity functions, using a parameter list, if available. The matching algorithm described at the end of Section 3.1 provides the archetypal example of the family of instance-based matching algorithms that the tool supports. In this example we used the similarity functions cosine distance and contrast model which are stored in Similarity. In step 1 we used the contrast model function applied to the token representation of properties (o). In step 2 we applied the contrast model function to the property representation of classes (props). In step 3 we used cosine distance function applied to token representation of instances (t). In step 4 we used contrast model function applied to token (o) and instance value (iv) representations of properties. The configuration of the similarity functions is stored by the Aplies class (top right of Figure 7).

Each execution of a matcher (bottom right of Figure 7) stores the parameter values that were used, and the similarity functions were applied (SimExecution in Figure 7). It also stores the order in which the similarity functions were applied, as well as values used in the computations. Each execution results in an aggregation of matching entries (Entry), which, in turn, model a vocabulary matching. In step 1 we used \( \alpha = 1.0, \beta = 3.5 \) and threshold (\( \tau \)) = max similarity – 31% as the parameters values. In step 2 we used threshold (\( \tau \)) = 0.8 as the parameter values. In this step, because we used the cosine distance, there were
no configuration parameters required to run the similarity function. In step 3 we used \( \alpha = 1.0, \beta = \gamma = 3.5 \) and threshold \( \tau = \text{max similarity} - 26\% \). In step 4 we used \( \alpha = 1.0, \beta = \gamma = 3.5 \) and threshold \( \tau = \text{max similarity} - 12\% \) for the similarities between token and instance representations of properties. All parameters were stored in the parameterValues of Execution. The type of an Entry tells us if it is false positive, false negative and so on. It is only used in cross validation matchings, as inputs for performance measuring. The similarity is the final similarity measure for each particular entry.

6 Conclusions

In this paper, we proposed hybrid matching techniques based on instance values and on schema information, such as datatypes, cardinality and relationships. The techniques uniformly apply similarity functions to generate matchings and are grounded on the interpretation, traditionally accepted, that “terms have the same extension when true of the same things” (Quine, 1968). In our context, two concepts match if they denote similar sets of objects. The techniques essentially differ on the nature of the sets to be compared and on the similarity functions adopted. For example and in a very intuitive way, two classes match if their sets of observed instances are similar, two terms from different thesauri match if the sets of instances they classify are similar, properties match if their sets of observed values are similar.

The assumptions that the database schemas we want to match are described in OWL notation, and that data from the databases can be obtained as sets of RDF triples facilitated the construction of matching techniques. However, the techniques introduced in the paper can be directly applied to conceptual schemas described in other conceptual modeling representation languages, such as the relational model (Codd, 1970). In conjunction, these assumptions permitted us to concentrate on a strategy to unveil the semantics of the database schemas to be matched, without being distracted by syntactical peculiarities. In fact, we consider good practice to provide OWL descriptions of the export schemas of data source providers. In conjunction with WSDL descriptions of the Web Services encapsulating the backend databases, this measure facilitates the interoperability of databases.

We focused on the more complex problem of matching two schemas that belong to an expressive OWL dialect. We decomposed the problem of OWL schema matching into the problems of vocabulary matching, and the problem of concept mapping. We also introduced sufficient conditions to guarantee that a vocabulary matching induces correct concept mappings. We adopted the contrast model (Tversky and Gati, 1978) as the preferred similarity function, which proved to efficiently capture the notion of similarity in this context, and described heuristics that led to practical OWL matchings.
Differently from the work of (Doan et al., 2001; Madhavan et al., 2005), we did not use machine learning techniques to acquire knowledge about matchings. Instead, we captured semantic similarity by adopting similarity functions and heuristics that depended on schema concepts. We consider this strategy to be more general because it identifies matching candidates that do not belong to the training corpus.

Unlike any of the instance-based techniques previously defined (see Section 1, the OWL schema matching process we described uses similarity functions to induce vocabulary matchings in a non-trivial way. The results demonstrated that the proposed technique performs well, with precision and recall rates around 80%.

Contrasting to the work of (Brauner et al., 2007b; Wang et al., 2004), which measure similarity between concepts based on the commonalities between sets of values alone, we made use of similarity functions that took into account not only the commonalities, but also the differences between concepts.

Differently from the work of (Bilke and Naumann, 2005), we overcame the limitations of representing instances using strings that concatenated all of its property values, by representing instances using strings that were constructed using only matching properties, as the first approximation.

As future work, we are considering three broad areas. First, further work is required on techniques to gradually construct the matchings as new data becomes available, which is typical of a query mediation environment. We refer the reader to (Brauner et al., 2006; Brauner et al., 2008) for discussions about this issue. Second, belief revision techniques should be investigated to help adjust the mediated schemas in time, as new data sources are integrated into the mediated environment. Third, implementation issues are pending, although (Gazola, 2008; Gazola et al., 2007) is a step in this direction.

In summary, unlike previous approaches, we proposed hybrid matching techniques that are uniformly grounded on similarity functions to generate matchings between simple catalogue schemas, as well as between more complex OWL schemas. We introduced the idea of decomposing the problem of vocabulary matching and concept mapping, which are often confused in the literature. We also showed when a vocabulary matching induces correct concept mappings, with respect to the integrity constraints of the schema, an issue also frequently overlooked in the literature.

Acknowledgements

This work was partly supported by CNPq under grants 557128/2009-9, 142103/2007-1, 301497/2006-0, and 473110/2008-3, and by FAPERJ under grant E-26/170028/2008.

References


