

# Instance-Based OWL Schema Matching

Luiz André P. Paes Leme, Marco A. Casanova  
Karin K. Breitman, and Antonio L. Furtado

Department of Informatics – Pontifical Catholic University of Rio de Janeiro  
Rua Marquês de S. Vicente, 225 – Rio de Janeiro, RJ – Brazil CEP 22451-900  
{lleme, casanova, karin, furtado}@inf.puc-rio.br

**Abstract.** Schema matching is a fundamental issue in 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. This paper describes an instance-based schema matching technique for an OWL dialect. The 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 functions.

## 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 a schema matching becomes a challenge when different vocabularies are used to refer to the same real-world concepts [6]. 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” [14].

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

The major contributions of this paper are three-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 a correct concept mapping. Second, we describe an OWL schema matching technique based on the notion of similarity. Third, we evaluate the precision of the technique using data available on the Web.

Rahm and Bernstein [15] is an early survey of schema matching techniques. Euzenat and Shvaiko [9] survey ontology matching techniques. Castano et al. [7] describe the H-Match algorithm to dynamically match ontologies.

Bilke and Naumann [1] describe an instance-based technique that explores similarity algorithms. Brauner et al. [2] adopt the same idea to match two thesauri. Wang et al. [16] describe a technique to match Web databases, which uses a set of typical instances. Brauner et al. [4] apply this idea to match geographical database. Brauner et al. [3] describe a matching algorithm based on measuring the similarity between attribute domains.

Unlike any of the above instance-based techniques, the matching process we describe uses similarity functions to induce vocabulary matchings in a non-trivial way, coping with an expressive OWL dialect. We also illustrate, through a set of examples, 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 schema matchings. Section 4 contains experimental results. Finally, Section 5 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 [5] for the 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  $p$ .

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 work with an OWL dialect, that we call *OWL Extralite*. It supports named classes, datatype and object properties, subclasses, and individuals. The *domain* of a datatype or object property is a class, the *range* of a datatype property is an XML schema type, whereas the *range* of an object property is a class. As *property restrictions*, the dialect admits `minCardinality` and `maxCardinality`, with the usual meaning. As *property characteristic*, it allows just the `InverseFunctionalProperty`, which captures simple keys. We note that only OWL Full supports the `InverseFunctionalProperty` for datatype properties.

An *OWL schema* (more often called an *OWL ontology*) is a collection of RDF triples that use the OWL vocabulary. A *concept* of 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). The scope of a property name is global to the OWL schema, and not local to the class indicated as its domain.

A triple of the form  $(s, \text{rdf:type}, c)$  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 an to OWL Extralite schema simply as a *schema*.

Figures 1 and 2 show schemas for fragments of the Amazon and the eBay databases, using a simplified 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, 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.

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.

Finally, to express concept mappings, we adopt the Semantic Web Rule Language (SWRL) [10], but use a datalog-like syntax to improve readability and save space. An example of an SWRL rule in our simplified syntax would be:

```
Product
title range string
listPrice range decimal
currency range string
Book is-a Product
author range string
edition range integer
isbn range string
ean range string
detailPageURL range anyURI
publisher range Publ
Publ
name range string
address range string
Music is-a Product
Video is-a Product
PCHardware is-a Product
```

**Fig. 1.** An OWL schema for a fragment of the Amazon Database

```
Seller
name range string
redistrationDate range dateTime
offers range Offer
Offer
quantity range integer
startPrice range double
currency range string
seller range Seller
product range Product
Product
title range string
condition range string
returnPolicyDetails range string
offers range Offer
Book is-a Product
author range string
edition range integer
publicationYear range integer
isbn-10 range integer
isbn-13 range integer
publisher range string
binding range string
condition range string
Music is-a Product
DVDMovies is-a Product
ComputerNetworking is-a Product
```

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

$$\text{eb:publisher}(b, n) \leftarrow \text{am:publisher}(b, p), \text{am:name}(p, n) \quad (1)$$

which says that, if  $b$  and  $p$  are related by  $\text{am:publisher}$ , and  $p$  and  $n$  by  $\text{am:name}$ , then  $b$  and  $n$  are related by  $\text{eb:publisher}$ .

## 2.2 Vocabulary Matchings and Concept Mappings

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  $(v_1, e_1, v_2, e_2)$  such that

- if  $(v_1, v_2) \in C_S \times C_T$ , then  $e_1$  and  $e_2$  are the top class  $\top$
- if  $(v_1, v_2) \in P_S \times P_T$ , then  $e_1$  and  $e_2$  are classes in  $C_S$  and  $C_T$  that must be subclasses of the domains, or the domains themselves, of properties  $v_1$  and  $v_2$ , respectively

If  $(v_1, e_1, v_2, e_2) \in \mu$ , we say that  $\mu$  *matches*  $v_1$  with  $v_2$  in the context of  $e_1$  and  $e_2$ , that  $e_i$  is the context of  $v_i$  and that  $(e_i, v_i)$  is a *contextualized concept*, for  $i=1,2$ .

Let  $\mathcal{Q}$  be an OWL query or rule language that supports the definition of classes and properties. In general, a *concept mapping* from  $S$  into  $T$  in  $\mathcal{Q}$  is a set  $\gamma$  of expressions of  $\mathcal{Q}$  that define concepts in  $T$  in terms of the concepts of  $S$ . Schemas  $S$  and  $T$  are called the *source* and the *target* of the concept mapping.

To detect when two instances denote the same real-world object, we need a third notion. Let  $U_S$  and  $U_T$  be sets of triples of  $S$  and  $T$ , respectively. An *instance matching* from  $S$  into  $T$  is a set  $\mu_i$  of quadruples such that, if  $(I, C, J, D) \in \mu_i$ , then there are triples  $(I, \text{rdf:type}, C) \in U_S$  and  $(J, \text{rdf:type}, D) \in U_T$ . We say that an instance  $I$  of a class  $C$  in  $U_S$  *matches* an instance  $J$  of a class  $D$  in  $U_T$  iff  $(I, C, J, D) \in \mu_i$ .

The following examples use the eBay and the Amazon schemas of Figures 1 and 2.

**Example 1.** Table 1 shows an example of a matching between the vocabularies of the eBay and the Amazon schemas. For example, line 1 indicates that classes  $\text{am:Book}$  and  $\text{eb:Book}$  match, in the sense that a triple  $(I, \text{rdf:type}, \text{am:Book})$  that defines  $I$  as an instance of  $\text{am:Book}$  may be reinterpreted as a triple  $(I, \text{rdf:type}, \text{eb:Book})$  that defines  $I$  as an instance of  $\text{eb:Book}$ .

Line 2 indicates that properties  $\text{am:title}$  and  $\text{eb:title}$  match only when their domains are restricted to  $\text{am:Book}$  and  $\text{eb:Book}$ , respectively. Thus, a triple  $(I, \text{am:title}, t)$  that defines  $t$  as a value of  $\text{am:title}$  may be reinterpreted as a triple  $(I, \text{eb:title}, t)$  that defines  $t$  as a value of  $\text{eb:title}$ , provided that  $I$  is an instance of  $\text{am:Book}$ .

In the next three examples, suppose that one wants to generate a concept mapping from the Amazon schema (the source schema) into the eBay schema (the target schema), using the vocabulary matching of Table 1.

**Table 1.** Example of a vocabulary matching

Amazon		eBay	
am:Book	⊤	eb:Book	⊤
am:title	am:Book	eb:title	eb:Book
am:author	am:Book	eb:author	eb:Book
am:listPrice	am:Product	eb:startPrice	eb:Offer
am:name	am:Publ	eb:publisher	eb:Book

**Example 2.** Line 1 of Table 1 indicates that `am:Book` matches `eb:Book`. It induces a mapping from `am:Book` into `eb:Book` expressed by the rule

$$\text{eb:Book}(n) \leftarrow \text{am:Book}(n) \quad (2)$$

From Figures 1 and 2, we have that `am:Book` is a subclass of `am:Product` and that `eb:Book` is a subclass of `eb:Product`. However, Table 1 does not indicate that `am:Product` matches `eb:Product`. Therefore, we must include an additional rule to guarantee that a consistent mapping is generated (see Section 2.3)

$$\text{eb:Product}(n) \leftarrow \text{am:Book}(n) \quad (3)$$

The mappings in (2) and (3) should be understood as follows. Let  $Q$  be a query over the eBay schema and assume that  $Q$  refers to `eb:Book`. Then,  $Q$  will be partly translated to the Amazon schema by replacing `eb:Book` by `am:Book`. Likewise, if  $Q$  refers to `eb:Product`, then  $Q$  will be partly translated by replacing `eb:Product` by `am:Book`. This means that, if  $Q$  asks for products, the translated query  $Q'$  will return only books.

**Example 3.** Consider line 2 of Table 1. Since `am:Book` is the context of `am:title`, the following rule expresses a correct mapping from `am:title` into `eb:title`

$$\text{eb:title}(b,n) \leftarrow \text{am:title}(b,n), \text{am:Book}(n) \quad (4)$$

The mapping in (4) should be understood as follows. Let  $Q$  be a query over the eBay schema and assume that  $Q$  refers to `eb:title`. Then,  $Q$  will be partly translated to the Amazon schema by replacing `eb:title` by

$$\text{am:title}(b,n) \text{ and } \text{am:Book}(n)$$

This means that, if  $Q$  asks for product titles, for example, the translated query  $Q'$  will return only book titles from the Amazon schema, since  $Q'$  has an extra restriction

$$\dots \text{and } \text{am:Book}(n)$$

Note that the right-hand side of the rule in (4) does not contain the context `eb:Book`, which will only be used when creating a concept mapping from the eBay schema into the Amazon schema.

**Example 4.** From Table 1 and Figures 1 and 2, we have:

$$\text{am:name matches eb:publisher} \quad (5)$$

$\text{am:Publ}$  and  $\text{eb:Book}$  are the domains of  $\text{am:name}$  and  $\text{eb:publisher}$  (6)

$\text{am:Publ}$  does not match  $\text{eb:Book}$  (7)

$\text{am:Book}$  matches  $\text{eb:Book}$  (8)

From (6), we cannot directly map  $\text{am:name}$  into  $\text{eb:publisher}$ . Indeed, the rule

$$\text{eb:publisher}(b, n) \leftarrow \text{am:name}(b, n) \quad (9)$$

expresses an incorrect mapping since  $b$  on the right-hand side stands for an instance of  $\text{am:Publ}$  (the domain of  $\text{am:name}$ ), whereas  $b$  on the left-hand side stands for an instance of  $\text{eb:Book}$  (the domain of  $\text{eb:publisher}$ ), but Table 1 does not indicate that  $\text{am:Publ}$  matches  $\text{eb:Book}$ . By contrast, the rule

$$\text{eb:publisher}(b, n) \leftarrow \text{am:publisher}(b, p), \text{am:name}(p, n) \quad (10)$$

is a correct mapping. Observing the right-hand side of the rule, we have that  $b$  stands for an instance of  $\text{am:Book}$ , which the object property  $\text{am:publisher}$  associates with an instance  $p$  of  $\text{am:Publ}$ , and the datatype property  $\text{am:name}$  in turn associates  $p$  with a string  $n$ . Now, observing the left-hand side of the rule, the datatype property  $\text{eb:publisher}$  associates  $b$ , an instance of  $\text{am:Book}$ , reinterpreted as an instance of  $\text{eb:Book}$ , (the domain of  $\text{eb:publisher}$ ) with  $n$ . This reinterpretation is consistent, since Table 1 also indicates that  $\text{am:Book}$  matches  $\text{eb:Book}$  (see Example 1).

### 2.3 Consistent OWL Matchings

We briefly discuss in this section the consistency of OWL Extralite vocabulary matchings, referring the reader to [12] 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  $(c_1, c_2, \dots, c_n)$  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  $c_{n-1}$  subsumes  $c_n$ . We consider that a class dominates itself.

A contextualized vocabulary matching  $\mu$  from  $S$  into  $T$  is *structurally correct* iff, for all  $(v_1, e_1, v_2, e_2) \in \mu$  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  $e_1$  (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  $\bar{\gamma}$  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, the property characteristics, the cardinality restrictions, and the 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 matching 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  $J = \bar{\gamma}(I)$  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 [12] for the details).

### 3 Instance-Based OWL Schema Matching

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

We first recall the matching technique for catalogue schemas based on similarity heuristics introduced in [11]. 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.

We also recall that the instance matching technique of Bilke and Naumann [1] represents each database tuple as a character string and uses k-mean clustering algorithms to find duplicate tuples. However, we note that the representations of the same object in distinct databases may differ in the list of attributes and in the attribute values. As a consequence, we may end up with dissimilar tuples that represent the same object.

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

eBay	Amazon
isbn-10 = "039577537X"	isbn = "039577537X"
isbn-13 = 9780395775370	ean = 9780395775370
title = "The Tragedy of Romeo and Juliet"	title = "Tragedy of Romeo and Juliet: And Related Readings (Literature Connections)"
author = "William Shakespeare"	author = "William Shakespeare"
publisher = "Houghton Mifflin"	name = "Houghton Mifflin Company"
returnPolicyDetails = "NO RETURNS ARE ACCEPTED"	-
condition = "Like New"	-
binding = "Hardcover"	-
-	listPrice = 18.92
-	currency = "USD"

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 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 the properties that match, the similarity increases to 70%. However, note that, to improve the instance matching strategy, we used the fact that `am:Book` matches `eb:Book`, and the fact that several 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) a class matching; and (b) an instance matching.
- (3) Use the class matching and the instance matching obtained in Step (2) to generate a refined contextualized property matching.
- (4) The final vocabulary matching is the result of the union of the class matching obtained in Step (2) and the property matching obtained in Step (3), adjusted until it becomes structurally correct.

Step (1) generates preliminary property matchings based on the intuition that "two properties match iff they that 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". However, to work correctly, Step (2) requires that Step (1) generates preliminary property matchings only for highly similar properties.

For example, in the experiments described in Section 4, with data from the eBay and the Amazon databases, if we use a threshold  $\tau=0.12$ , then `eb:level` with context `eb:Seller` matches `am:color` with context `am:PCHardware` and `eb:title` with context `eb:Music` matches `am:title` with context `am:Video`. These property matchings may cause classes `eb:Seller` and `am:PCHardware` to match, as well as

eb:Music and am:Video, depending on the threshold and the total amount of common properties among the classes (as discussed below, class matching depends on the similarity between sets of properties). If we increase the threshold to 0.13, the previous property matchings do not hold, avoiding the above unwanted class matchings.

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.

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  $o[U_S, P^C]$  of all  $v$  such that there are triples of the form  $(I, P, v)$  and  $(I, \text{rdf:type}, C')$  in  $U_S$ , 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 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

$$(P, C, Q, D) \in \mu_P \text{ iff } \sigma(o[U_S, P^C], o[U_T, Q^D]) \geq \tau \quad (11)$$

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 *representation* of  $C$  in  $U_S$ .

The *contextualized class matching 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  $\top$  is the top class)

$$(C, T, D, \top) \in \mu_C \text{ iff } \sigma(\text{props}[S, C], \text{relprops}[S, C, T, D]) \geq \tau \quad (12)$$

where  $\text{relprops}[S, C, T, D]$  denotes the set of properties  $P$  of class  $C$  of  $S$  such that there is a property  $Q$  of class  $D$  of  $T$  such that  $(P, C, Q, D) \in \mu_P$ . 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  $\text{relprops}[S, C, T, D]$ .

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

Figure 3 shows the algorithm that computes the instance matching. It receives as input  $S$  and  $T$ , and the class matching  $\mu_C$  induced by  $\sigma$ ,  $\tau$  and  $\mu_P$ . It also implicitly receives as input  $U_S$  and  $U_T$ . It outputs an instance matching  $\mu_I$  between class instances in  $U_S$  and  $U_T$ . In Figure 3, if  $C$  is a class in  $C_S$ , and  $I$  is an instance of  $C$  in  $U_S$ , then  $t[U_S, C](I)$  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  $\mu_P$ , and likewise for  $t[U_T, D](J)$ .

Figure 4 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  $(C, T, D, \top) \in \mu_C$ , 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  $U_S$ , there are triples of the form  $(J, Q, v)$  and

```

INSTANCE-MATCHING(S,T,μC)
for each pair of classes (C,D) in S and T
  such that μC matches C with D
  for each pair of instances (I,J) of C and D in US and UT
    if σ(t[US,C](I),t[UT,D](J)) ≥ τ
      then μI = μI ∪ (I,C,J,D)

```

Fig. 3. The class instance matching algorithm

```

CONTEXTUALIZED-PROPERTY-MATCHING(S,T,μC)
for each pair of classes (C,D) in S and T
  such that μC matches C with D
  or C' dominates C and μC matches C' with D
  or μC matches C with D' and D' dominates D
  for each pair (P,Q) of properties of C and D
    X = σ(o[US,PC],o[UT,QD])
    if (C matches D)
      then (s,t) = iv[P,C,Q,D]
    Y = σ(s,t)
    else Y = 0
    if max(X,Y) ≥ τ'
      then μA = μA ∪ (P,C,Q,D)

```

Fig. 4. The contextualized property matching algorithm

$(J, \text{rdf:type}, D)$  in  $U_T$ , and  $(I, C, J, D) \in \mu_I$  (where  $\mu_I$  is the instance matching of Figure 3). Define  $iv[P, C, Q, D] = (s, t)$  such that  $s = \{(I, u) \mid (\exists v)(I, u, v) \in q\}$  and  $t = \{(I, v) \mid (\exists u)(I, u, v) \in q\}$ . We call  $s$  the *instance-value representation* of  $P^C$  in  $U_S$  (and likewise for  $t$ ). This second representation is useful since it helps distinguish properties with similar sets of values, but which refer to distinct instances, matched by  $\mu_I$ .

Returning to the algorithm in Figure 4, it has the same input as the algorithm in Figure 3, and outputs a contextualized property matching  $\mu_A$  only between properties whose domains are classes directly or indirectly matched by  $\mu_C$ . The algorithm uses the maximum of the 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 4, object properties receive a special treatment, since their representations are sets of URIs that are compared with help of the instance matching  $\mu_I$  (computed by the algorithm in Figure 3).

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  $\mu_A$  computed by the algorithm in Figure 4. However,  $\mu$  may have to be adjusted, by dropping matchings, until it becomes structurally correct (details omitted for brevity).

## 4 Experimental Results

We conducted an experiment to assess the performance of the vocabulary matching process of Section 3, using data about products obtained from Amazon and eBay.

We tested the process with data downloaded from the Web, rather than with the benchmark proposed in Duchateau et al [8], since the benchmark does not include instances and is therefore unsuitable to test our process.

**Table 3.** Automatically obtained vocabulary matching from eBay into Amazon

#	eBay		Amazon		Match Type
	v1	e1	v2	e2	
1	Books	T	Books	T	tp
2	author	B	author	B	tp
3	edition	B	edition	B	tp
4	format	B	biding	B	tp
5	isbn-10	B	isbn	B	tp
6	isbn-13	B	ean	B	tp
7	editionDesc	B	format	B	fp
8	Offer	T	Books		fp

We first defined a set of terms, which were used to query the databases. From the query results, we extracted the less frequent terms common to both databases. We then used these terms to once more query the databases. 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.

We adopted as similarity functions the contrast model [11], for property matchings, and the cosine distance with TF/IDF, for instance matchings. The experiments lead us to conclude that the contrast model has a better performance when we want to emphasize the difference between two sets of values. This follows because the contrast model has room for calibrating several parameters.

Table 3 shows sample entries of the vocabulary matching obtained. The headings indicate that **e1** is the context of **v1**, and **e2** that of **v2**. Also, “B” abbreviates classes `eb:Book` and `am:Book`.

The rightmost column of Table 3 classifies the matchings: *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:

$$precision = \frac{tp}{tp + fp} = 86\% \quad recall = \frac{tp}{tp + fn} = 71\% \quad fMeasure = 2 \frac{precision \cdot recall}{precision + recall} = 78\%$$

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:

$$precision = 82\% \quad recall = 51\% \quad fMeasure = 63\%$$

## 5 Conclusions

In this paper, we proposed a process to match the vocabularies of pairs of OWL extralite schemas and to create a concept mapping out of a vocabulary matching. The process is instance-based and uses similarity functions to induce vocabulary matchings in a non-trivial way. The last step of the process guarantees that the final vocabulary matching is structurally correct and, therefore, induces a consistent concept mapping. We illustrated the approach with experiments using data available on the Web.

The results described in the paper admit several extensions. In particular, we may extend the process to gradually revise the matchings as new data becomes available, which is typical of a query mediation environment. We may also extend the process to more complex OWL schemas, which requires a strategy to revise the target OWL schema.

**Acknowledgements.** This work was partly supported by CNPq under grants 142103/2007-1, 301497/2006-0 and 473110/2008-3.

## References

1. Bilke, A., Naumann, F.: Schema matching using duplicates. In: Proc. of the 21st Int'l. Conf. on Data Engineering, pp. 69–80
2. Brauner, D.F., Casanova, M.A., Milidiú, R.L.: Towards Gazetteer Integration Through an Instance-based Thesauri Mapping Approach. In: Adv. in Geoinformatics, pp. 235–245. Springer, Heidelberg
3. Brauner, D.F., Gazola, A., Casanova, M.A.: Adaptative matching of database web services export schemas. In: Proc. of the 10th Int'l. Conf. on Enterprise Inf. Systems
4. Brauner, D.F., Intrator, C., Freitas, J.C., Casanova, M.A.: An instance-based approach for matching export schemas of geographical database Web services. In: Proc. of the IX Brazilian Symp. on GeoInformatics (GeoInfo), pp. 109–120
5. Breitman, K., Casanova, M., Truskowski, W.: Semantic web: concepts, technologies, and applications. Springer, London
6. Casanova, M., Breitman, K., Brauner, D., Marins, A.: Database conceptual schema matching. *Computer* 40(10), 102–104
7. Castano, S., Ferrara, A., Montanelli, S., Racca, G.: Semantic Information Interoperability in Open Networked Systems. In: Proc. ICSNW, in cooperation with ACM SIGMOD 2004, Paris, France (2004)
8. Duchateau, F., Bellahsene, Z., Hunt, E.: XBenchMatch: a benchmark for XML schema matching tools. In: Proc. 33th Int'l. Conf. on VLDB, Demo Sessions, pp. 1318–1321
9. Euzenat, J., Shvaiko, P.: *Ontology matching*. Springer, Heidelberg
10. Horrocks, I., Patel-Schneider, P.F., Boley, H., Tabet, S., Groszofand, B., Dean, M.: SWRL: A semantic web rule language combining OWL and RuleML. W3C
11. Brauner, D.F., Breitman, K.K., Casanova, M.A., Gazola, A.: Matching object catalogues. *J. Innovations in Systems and Software Engineering* 4(4), 315–328
12. Leme, L.A.P.P.: *Conceptual schema matching based on similarity heuristics*. D.Sc. Thesis, Dept. Informatics, PUC-Rio

13. Leme, L.A.P.P., et al.: Evaluation of similarity measures and heuristics for simple RDF schema matching. Technical Report 44/08, Dept. Informatics, PUC-Rio
14. Quine, W.V.: Ontological Relativity. *J. of Philosophy* 65(7), 185–212
15. Rahm, E., Bernstein, P.: A survey of approaches to automatic schema matching. *The VLDB Journal* 10(4), 334–350
16. Wang, J., Wen, J., Lochovsky, F., Ma, W.: Instance-based schema matching for web databases by domain-specific query probing. In: *Proc. 13th Int'l. Conf. on VLDB*, pp. 408–419.