PUC

ISSN 0103-9741

Monografias em Ciência da Computação
n° 44/08

Evaluation of Similarity Measures and Heuristics for Simple RDF Schema Matching

Luiz André Portes Paes Leme

Marco Antonio Casanova

Karin Koogan Breitman

Antonio L. Furtado

Departamento de Informática

PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO DE JANEIRO

RUA MARQUÊS DE SÃO VICENTE, 225 - CEP 22451-900

RIO DE JANEIRO - BRASIL

Monografias em Ciência da Computação, No. 44/08 ISSN: 0103-9741
Editor: Prof. Carlos José Pereira de Lucena October, 2008

Evaluation of Similarity Measures and Heuristics for Simple RDF Schema Matching *

Luiz André Portes Paes Leme, Marco Antonio Casanova,

Karin Koogan Breitman, Antonio L. Furtado

{lleme, casanova, karin, furtado}@inf.puc-rio.br

Abstract. Schema matching is a fundamental issue in database applications, such as query mediation and data warehousing. In this paper, we assume that each database schema to be matched is described in RDF, and contains only class definitions and property definitions whose ranges are XML Schema simple types. We propose and compare RDF property matching heuristics based on similarity functions, applied to sets of observed values. We describe experimental results that show that customized contrast models induce good quality RDF property matchings.

Keywords: database, matching, schema, similarity.

Resumo. Alinhamento de esquema é uma questão fundamental em aplicações de banco de dados, tais como, mediação de consultas e armazéns de dados. Neste trabalho, assumimos que os esquemas de bancos de dados a serem alinhados são descritos em RDF e contém somente definições de classes e propriedades cujos tipos são tipos simples do esquema XML. Propomos e comparamos heurísticas de alinhamento baseadas em funções de similaridade aplicadas a conjuntos de valores observados das propriedades. Descrevemos, também, resultados experimentais que mostram o bom desempenho de um modelo de similaridade adaptado a partir do *contrast model*.

Palavras-chave: banco de dados, alinhamento, esquema, similaridade.

* This work has been sponsored by CNPq under grants 142103/2007-1 and 301497/2006-0, 550930/2007-8.

Table of Contents

1 Introduction

Schema matching is a fundamental issue in database applications, such as query mediation and data warehousing. A reasonable approach to schema matching, sometimes called extensional, instance-based or semantic, is to detect how the same real-world objects are represented in different databases, to compare property values and to use the information thus obtained to match the export schemas. Such approach is more robust than purely syntactical approaches, but it applies only when the schemas to be matched are simple.

In this paper, we assume that each database schema to be matched is described in RDF, and contains only class definitions and property definitions whose ranges are XML Schema simple types. We also assume that the data obtained from the databases is available as sets of RDF triples. The first assumption avoids the complications of dealing with more complex constructs, such as class and property hierarchies, object properties (in OWL jargon) and complex XML schema types. It should be viewed as a recommendation for those designing Web services that encapsulate databases to be freely available over the Web. The second assumption avoids the burden of interpreting the format of data exported from the databases. In conjunction, these assumptions permit us to concentrate on a strategy to unveil the semantics of the database schemas to be matched, without being distracted by syntactical peculiarities.

A schema matching between two schemas R and S relates classes of R with classes of S and properties of R with properties of S. The problem of defining a schema matching between R and S breaks down into defining a class matching and defining a property matching. Furthermore, an instance matching between R and S is a relation between RDF triples of R and RDF triples of S. In this paper, we focus on extensional property matching techniques.

The major contributions of this paper are three-fold. First, we give a precise definition of RDF schema matching based on extensional techniques. Second, we develop a similarity function based on contrast models [Tversky 1977], which proved to efficiently capture the notion of similarity, and describe heuristics that lead to practical RDF property matchings. Third, we describe experimental results that evaluate the precision of the RDF property matchings introduced, measure the influence of the heuristics and compare the customized contrast model with three different similarity functions.

This paper is organized as follows. Section 2 provides a very brief overview of RDF, following [Breitman, Casanova e Truszkowski 2007], and introduces additional concepts required in the rest of the paper. Section 3 describes in detail the similarity functions proposed in the paper. Section 4 contains experimental results. Section 5 summarizes related work. Finally, section 6 contains the conclusions and directions for future research.

2 Background

2.1 A brief introduction to RDF

A resource is anything that has an identity, be it a retrievable digital entity (such as an electronic document, an image, or a service), a physical entity (such as a book) or a collection of other resources. A Uniform Resource Identifier (URI) is a character string that identifies an abstract or physical resource on the Web. A URI reference (URIref) denotes the common usage of a URI, with an optional fragment identifier attached to it and preceded by the character "#".

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 *(also called the* predicate*) of the statement, that denotes a binary relationship*
- 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 the property* P

RDF offers enormous flexibility but, apart from the rdf:type property, which has a predefined semantics, it provides no means for defining application-specific classes and properties. Instead, such classes and properties, and hierarchies thereof, are described using extensions to RDF provided by the RDF Vocabulary Description Language 1.0: RDF Schema (RDF Schema or RDF-S).

In RDF Schema, a class is any resource having an rdf:type property whose value is the qualified name rdfs:Class of the RDF Schema vocabulary.

A class C is defined as a subclass of a class D by using the predefined rdfs:subClassOf property to relate the two classes. The rdfs:subClassOf property is transitive in RDF Schema.

A property is any instance of the class rdfs:Property. The rdfs:domain property is used to indicate that a particular property applies to a designated class, and the rdfs:range property is used to indicate that the values of a particular property are instances of a designated class or, alternatively, are instances (i.e., literals) of an XML Schema datatype.

The specialization relationship between two properties is described using the predefined rdfs:subPropertyOf sub. An RDF property may have zero or more sub properties; all RDF Schema rdfs:range and rdfs:domain properties that apply to an RDF property also apply to each of its sub properties.

An instance of a class C is a resource I having an rdf:type property whose value is C, which is indicated by the RDF statement (I, rdf:type, C). A resource may be an instance of more than one class. To define that an instance I of a class has a property P with value V, we simply define an RDF statement (I, P, V).

We introduce a simple RDF database schema as a set R of triples in the RDF Schema vocabulary that define only classes and properties, and property ranges that are XML Schema simple types. A simple RDF database schema therefore does not define subclasses, subproperties, properties whose ranges are classes (i.e., object properties in OWL) and instances.

An RDF triple is of R iff it defines an instance of a class defined in R or the value of a property defined in R.

An observed extension for R is a set oR of RDF triples of R. The set of observed values of a property P of R in oR is defined as

$$o_R[P] = \{ V \,/\, (S,P,V) \in o_R \}$$

Likewise, the set of observed instances of a class C of R in oR is defined as

$$o_R[C] = \{ S \,/\, (S,\text{rdf:type},C) \in o_R \}$$

2.2 Similarity-induced matching

In what follows, let R and S be two simple RDF database schemas, whose sets of classes and properties are C[R] and P[R], and C[S] and P[S], respectively.

A *schema matching* between *R* and *S* is a partial, many-to-many relation $\mu_a \subseteq C[R] \times C[S] \cup P[R] \times P[S]$ that relate classes of *R* with classes of *S* and properties of *R* with properties of *S*. We allow μ_a to be partial since some class or property of *R* may not match any class or property of *S* and we let μ_a to be many-to-many to account for classes or properties from *R* that match several classes or properties of *S*, and vice-versa (in Section 4 the eBay schema, exemplifies this situation). We say that μ_a is *unambiguous* iff μ_a is one-to-one.

Note that the problem of defining a schema matching between *R* and *S* naturally breaks down into two problems: defining a *class matching* and defining a *property matching*.

An *instance matching* between *R* and *S* is a partial, many-to-many relation μ_i between RDF triples of *R* and RDF triples of *S*.

In this paper, we are interested in extensional schema matching techniques that use duplicated values to formulate hypothesis about the matching. More specifically, we focus on extensional property matching techniques.

Briefly, let *P* and *Q* be two properties, and *C* and *D* be two classes from *R* and *S*, respectively. Let o_R and o_S be observed extensions of *R* and *S*.

We may postulate that *P* and *Q* match based on the similarity between the sets of observed values for *P* and *Q* in o_R and o_S, respectively. Likewise, we may postulate that *C* and *D* match based on the similarity between the sets of observed instances for *C* and *D* in o_R and o_S, respectively. The question of obtaining a schema matching between *R* and *S* then reduces to finding reasonable similarity functions between sets of values and between sets of instances, in the presence of observed extensions for *R* and *S*.

Similarly, let *I* and *J* be observed instances of classes *C* and *D* in o_R and o_S, respectively. We may postulate that *I* and *J* match iff they are similar to each other, in the context of the observed extensions o_R and o_S.

In this paper, we are interested in investigating similarity functions that induce good property matchings. Section 3.1 introduces several similarity functions, whereas section 3.2 argues that it may sometimes be advantageous not to apply the similarity functions directly to the sets of observed values, but rather to other sets derived from them. Section 4 introduces the notion of similarity model that leads to property matchings and contains experimental results.

We conclude this section with two practical considerations. First, computing property similarities based on observed extensions depends on finding a sufficiently large number of duplicates from the RDF sources. Second, the hypothesis of knowing all observable values is unrealistic for obvious reasons. Therefore, the matching approach should be understood as an incremental process where the results of user queries are accumulated and continuously analyzed until one obtains a stable model, as discussed in [Brauner, Casanova e Milidiú 2006].

3 Proposed property matching technique

3.1 Similarity functions

Similarity is a concept frequently used in many different applications. Different similarity functions have been proposed over the years such as information content [Resnik 1995], information theory [Lin 1998] [Brauner, Gazola e Casanova 2008] [Hindle 1990], vector model [Frakes e Baeza-Yates 1992], distance measurements [Lee 1993] and the contrast model [Tversky 1977].

In this paper we compare results of attribute matchings using four similarity functions. The first one is based on the vector model. In text processing applications [Frakes e Baeza-Yates 1992], each document is represented by a vector $\vec{A}$, where each dimension represents an index term and the coordinate is the weight of the term in the document. Similarly, a user query is represented by another vector $\vec{B}$ with the same number of dimensions, but with the coordinate representing the weight of the index term in the query. The vector model proposes to evaluate the degree of similarity of a document with regard to a query as the correlation between the vectors. This correlation can be quantified, for instance, by the cosine of the angle between these two vectors. That is,

$$sim(\vec{A},\vec{B}) = \frac{\vec{A} \bullet \vec{B}}{|\vec{A}||\vec{B}|}$$

In the following sections we will extend the concepts of *index vectors* and *weights* in the context of attribute matching.

The second similarity function is based on Information Theory and was proposed in [Lin 1998]. According to the authors, the similarity between two objects *A* and *B* is a function of the amount of information in the propositions of their *commonalities* and *descriptions*. They use the conclusion of Cover and Thomas [Cover e Thomas 1991] which says that the information contained in a statement is measured by the negative logarithm of the probability of the statement. Imposing some restrictions and algebraically manipulating the function, the similarity between two objects is define as

$$sim(A,B) = \frac{\log(P(commonalities(A,B)))}{\log(P(description(A,B)))}$$

where *commonalities(x,y)* and *description(x,y)* are functions that return subsets of features of the objects *x* and *y* and *P(x)* is the probability of the set of features *x*. For example, if *A* is an orange and *B* is an apple, the *commonalities* between *A* and *B* can be state as the proposition *fruit(A) and fruit(B).* The predicate *fruit* is a possible common feature to both objects from a predefine set of features. If *A* and *B* belong to a set *S* of objects then

$$P(\text{commonalities}(A,B)) = P(\text{fruit}(x) \text{ and } \text{fruit}(y)) \mid x,y \in S$$

i.e. the probability of commonalities is the probability of the occurrence of the predicate *fruit* on two objects. The description of *A* and *B* is the probability of union of all features of the two objects.

The third similarity function is presented in [Brauner, Gazola e Casanova 2008], where the authors address the problem of matching the attributes of two relational schemes, $R[A_1,\ldots,A_m]$ and $S[B_1,\ldots,B_m]$. Given two relations σ_R and σ_S that follow the schemes *R* and *S*, the authors first propose to compute the m×n co-occurrence matrix $[m_{ij}]$ such that m_{ij} is the cardinality of $\sigma_R[A_i] \cap \sigma_S[B_j]$. The next step is to compute the *Estimated Mutual Information* matrix EMI defined as

$$EMI(A_r, B_s) = \frac{m_{rs}}{\sum_{i,j} m_{ij}} \log \left(\frac{\dfrac{m_{rs}}{\sum_{i,j} m_{ij}}}{\dfrac{\sum_j m_{rj}}{\sum_{i,j} m_{ij}} * \dfrac{\sum_i m_{is}}{\sum_{i,j} m_{ij}}} \right),$$

The authors then postulate that two attributes A_r and B_s match iff $EMI(A_r,B_s) \geq EMI(A_r,B_j)$, for all $j \in [1,n]$, with $j \neq s$, and $EMI(A_r,B_s) \geq EMI(A_i,B_s)$, for all $i \in [1,m]$, with $i \neq r$.

The last similarity function is based on the contrast model [Tversky 1977], which states that the similarity between *x* and *y* increases with the amount of features, measured by a given function *f*, which *x* and *y* have in common, and decreases with the amount of features which belong to just *x* or to just *y*. The notion of feature is used here with the same meaning as in the second similarity function presented before, i.e. they are predicates or characteristics of objects. The contrast model has been evaluated and successfully used in many applications [Eidenberger 2006] [Eidenberger e Breiteneder 2002] [Tang et al. 2007]. One possible reason for its success is that it is very close to the human perception of similarity.

More precisely, let C be a set of features and let 2^C denote the power set of C. Let $f{:}2^C \rightarrow \mathbb{R}^+$ be a *scale function* for C. A *contrast model* is a function $\tau{:}2^C \times 2^C \rightarrow \mathbb{R}^+$ such that

(1) $$\tau(x,y) = \theta f(x \cap y) - \alpha f(x-y) - \beta f(y-x)$$

for any $x, y \in 2^C$, where $\theta, \alpha, \beta \in \mathbb{R}^+$ are the *parameters* of the contrast model. Note that this formula defines a class of models that depend on the choice of f, θ, α and β.

Now, let $N{:}\ 2^C \rightarrow \mathbb{R}^+$ denote a function that takes a set *x* of values and returns its cardinality $|x|$. Using this function as the scale function, we may successively rewrite (1) as:

(2) $$\tau_{\theta,\alpha,\beta}(x,y) = \theta N(x \cap y) - \alpha N(x-y) - \beta N(y-x)$$

(3) $$\tau_{\theta,\alpha,\beta}(x,y) = \theta N(x \cap y) - \alpha(N(x) - N(x \cap y)) - \beta(N(y) - N(x \cap y))$$

(4) $$\tau_{\theta,\alpha,\beta}(x,y) = (\theta + \alpha + \beta)N(x \cap y) - \alpha N(x) - \beta N(y)\text{, for any } x, y \in 2C$$

In order to normalize the result, Equation (4) can be balanced with $|C|$, the cardinality of C. We then redefine the function $\tau_{\theta,\alpha,\beta}$ as

(5) $$\tau_{\theta,\alpha,\beta}(x,y) = \frac{(\theta+\alpha+\beta)N(x\cap y) - \alpha N(x) - \beta N(y)}{|C|}$$

To simplify the notation, define $\overline{N}(x) = N(x)/|C|$ and rewrite equation (5) as:

(6) $$\tau_{\theta,\alpha,\beta}(x,y) = (\theta+\alpha+\beta)\overline{N}(x\cap y) - (\alpha\overline{N}(x) + \beta\overline{N}(y))$$

The image of such function is contained in $\mathbb{R}^+$, which imposes serious restrictions on fixing a threshold to select similar properties. For this reason, it is convenient to rewrite the formula using *log(*$\overline{N}(x)$*)*, instead of $\overline{N}(x)$.

(7) $$\tau_{\theta,\alpha,\beta}(x,y) = \log\left(\frac{\overline{N}(x\cap y)^{(\theta+\alpha+\beta)}}{\overline{N}(x)^{\alpha}\overline{N}(y)^{\beta}}\right)$$

Since $\overline{N}(x\cap y)\geq 0$, $\overline{N}(x)\geq 0$, $\overline{N}(y)\geq 0$, $\overline{N}(x\cap y)\leq\overline{N}(x)$, $\overline{N}(x\cap y)\leq\overline{N}(y)$, $\overline{N}(x\cap y)\leq 1$, then $\tau_{\theta,\alpha,\beta}(x,y)$ is always negative.

In order to limit the similarity values to the interval [0.0,1.0] equation (7) can be rewritten as

(8) $$\tau_{\theta,\alpha,\beta}(x,y) = \left(1 - \log\left(\frac{\overline{N}(x\cap y)^{(\theta+\alpha+\beta)}}{\overline{N}(x)^{\alpha}\overline{N}(y)^{\beta}}\right)\right)^{-1}$$

3.2 Property matching heuristics

Let R and S be two simple RDF database schemas, whose sets of properties are $P[R]$ and $P[S]$, respectively. Let $A\in P[R]$ and $B\in P[S]$. Let o_R and o_S be observed extensions of R and S. Recall that $o_R[A]$ denotes the set of observed values of A in o_R (and likewise for $o_S[B]$).

Let U denote the universe of all XML Schema type values.

Consider a (generic) similarity function $\sigma: 2^U\times 2^U\rightarrow\mathbb{R}^+$ over 2^U and a positive Real number, s, the *similarity threshold* for sets of values in U. Define the property matching between R and S *induced by σ and s for o_R and o_S* as the partial, many-to-many relation $\mu\subseteq P[R]\times P[S]$ such that $(A,B)\in\mu$ iff $\sigma(o_R[A],o_S[B])\geq s$.

We discuss three practical heuristics to compute or redefine μ. Note that the discussion in principle applies to any similarity function over 2^U, and not just to the contrast model.

The first heuristic, called the *type compatibility heuristic*, is quite simple: we compute $\sigma(o_R[A],o_S[B])$ only if A and B are of the same type (or whose type is compatible). This heuristic is advantageous since it avoids testing all $(m\times n)/2$ possible combinations of properties from R with properties from S, assuming that the similarity function is symmetric.

The next three heuristics actually redefine the notion of property matching from R into S induced by σ and s by applying σ to sets derived from the sets of observed values.

The second heuristic, called the *multiset domain heuristic*, is to consider the *multiset of observed values* of a property A, defined as the multiset that contains as many elements corresponding to a single value v as the number of triples whose value for A is v. Intuitively, the motivation for this heuristic is to take into account, in the similarity function, the number of times a value occurs for a property. The results in Section 4 suggest that this heuristic is actually not very effective.

Suppose that *A* is of type *string*. The third heuristic, called the *string domain heuristic*, is to consider the *set of observed tokens* extracted from the strings in $o_R[A]$. This set is obtained as follows. First, tokens are extracted from a string *str* by splitting *str* in each non-word or non-numeric characters to obtain a set of substrings from *str*. Then, this set is reduced by eliminating substrings which are stop-words. Finally, the remaining strings are lemmatized [Manning e Schütze 2002]. This heuristic facilitates the application of the similarity function σ to pairs of properties of type *string*.

The fourth heuristic, called the *instance matching heuristic*, is not so obvious. To explain it, consider two simple RDF database schemas: R_1, with a single class B_1, properties $ISBN_1$, $Name_1$, $Edition_1$ and $Rating_1$ and extension o_{R1} (Fig. 1); and R_2, also with a single class B_2, properties $ISBN_2$, $Name_2$, $Edition_2$, $Rating_2$ and extension o_{R2}. Figure 1 illustrates possible extensions for R_1 and R_2.

Assume that the set of observed values of $ISBN_1$ and $ISBN_2$ are sets of 13-digit ISBNs, and that the sets of observed values of $Edition_1$, $Rating_1$, $Edition_2$ and $Rating_2$ are sets of small integers (book editions typically range from 1 to 10, and book ratings from 1 to 5, say). Figure 2 illustrates the sets of observed values for the properties of R_1 and R_2, considering the extensions shown in Figure 1.

Suppose that the correct property matchings are $ISBN_1$ with $ISBN_2$, $Edition_1$ with $Edition_2$ and $Rating_1$ with $Rating_2$.

o_{R1}				o_{R2}		
S	**P**	**V**		**S'**	**P'**	**V'**
10	$ISBN_1$	100		100	$ISBN_2$	100
10	$Edition_1$	1		100	$Edition_2$	1
10	$Rating_1$	2		100	$Rating_2$	2
10	$Name_1$	Book1		100	$Name_2$	Book1
20	$ISBN_1$	101		200	$ISBN_2$	101
20	$Edition_1$	2		200	$Edition_2$	2
20	$Rating_1$	3		200	$Rating_2$	3
20	$Name_1$	Book2		200	$Name_2$	Book2
30	$ISBN_1$	102		300	$ISBN_2$	103
30	$Edition_1$	3		300	$Edition_2$	3
30	$Rating_1$	4		300	$Rating_2$	4
30	$Name_1$	Book3		300	$Name_2$	Book4
40	$ISBN_1$	104		400	$ISBN_2$	104
40	$Edition_1$	4		400	$Edition_2$	4
40	$Rating_1$	1		400	$Rating_2$	1
40	$Name_1$	Book5		400	$Name_2$	Book5

Fig. 1 Extensions of R1 and R2.

$ISBN_1$	$ISBN_2$	$Edition_1$	$Edition_2$	$Rating_1$	$Rating_2$	$Name_1$	$Name_2$
100	100	1	1	1	1	Book1	Book1
101	101	2	2	2	2	Book2	Book2
102	103	3	3	3	3	Book3	Book4
104	104	4	4	4	4	Book5	Book5

Fig. 2 Sets of observed values for the properties of R1 and R2.

Then, the similarity function σ may induce a correct match between $ISBN_1$ and $ISBN_2$, and $Name_1$ and $Name_2$, since these properties have sets of observed values which will tend to be similar to each other (Fig. 2), and very different from the sets of observed values of the other properties. However, the similarity function σ may not generate the other correct matchings since the properties involved may have about the same sets of observed values.

To circumvent this limitation, assume that, instead of considering the set of observed values of $Edition_1$, we consider the set IE_1 of pairs (i,e) such that there are triples $(s,ISBN_1,i)$ and $(s, Edition_1,e)$ in the extension o_{R1} and likewise for the other three properties, generating sets IE_2, IR_1 and IR_2 Figure 3 illustrates such sets for the properties of R_1 and R_2, considering the extensions shown in Figure 1.

Then, the similarity function σ has a better chance of distinguishing the correct matchings from the incorrect matchings. That is, $\sigma(IE_1, IE_2)$ and $\sigma(IR_1, IR_2)$ are likely to be higher than $\sigma(IE_1, IR_2)$ and $\sigma(IE_2, IR_1)$ (the other pairs need not be considered since σ is symmetric). Intuitively, an incorrect matching of IE_1 with IR_2 would be plausible only if a large number of books have the same edition number in B_1 as the rating value they have in B_2, which is less likely that a large number of books occurring with the same edition number in both B_1 and B_2 (and likewise for the other pairs).

ISBN₁ Edition₁	***ISBN₂ Edition₂***	***ISBN₁ Rating₁***	***ISBN₂ Rating₂***	***ISBN₁ Name₁***	***ISBN₂ Name₂***
(100,1)	(100,1)	(100,2)	(100,2)	(100,Book1)	(100,Book1)
(101,2)	(101,2)	(101,3)	(101,3)	(101,Book2)	(101,Book2)
(103,3)	(103,3)	(103,4)	(103,4)	(103,Book4)	(103,Book4)
(104,4)	(104,4)	(104,1)	(104,1)	(104,Book5)	(104,Book5)

Fig. 3 Co-observed values of the properties of R1 and R2.

This heuristic may be reinterpreted as a combination of instance matching with property matching (hence its name). In general, let R, S, o_R, o_S, U, σ and s be as before. Let α be an instance matching between R and S. Define the property matching between R and S *induced by* σ, s and α *for* o_R *and* o_S as the partial, many-to-many relation $\mu \subseteq P[R] \times P[S]$ such that

$(A,B) \in \mu$ iff $\sigma(o_a,o_b) \geq s$

where

$o_a = \{ (I,U) / (I,A,U) \in o_R \}$

$o_b = \{ (I,V) / (J,B,V) \in o_S$ and $(I,J) \in \alpha \}$

In our running example, two instances of classes B_1 and B_2 are considered to match iff they have the same value for the properties $ISBN_1$ and $ISBN_2$.

Obviously, the instance matching heuristic requires that a proper instance matching be previously defined. However, section 4 shows that this heuristic also brings significant improvements when there is a sufficiently large number of instance matchings.

At this point, note that we actually introduced two notions of property matchings induced by similarity functions: one that applies the similarity function directly to sets of observed values; and another that uses instance matchings in the definition. Intuitively, the first approach seems to have the best performance when there is an insufficient number of instance matchings, whereas the second approach has a better performance, otherwise.

In addition, when trying to determine if A and B match, we may apply σ to the sets of observed values of A and B, to the multisets of observed values of A and B, to the set of observed tokens of A and B, etc… In fact, we may define a more complex strategy that computes several such possibilities to each set of values to decide if A and B match or not. Therefore, a generic *similarity model* should adopt the best approach, as discussed in Section 4.

For example, in our running example concerning books, we may define a similarity model that, for each pair of properties:

- computes the similarity between their multiset of observed values;
- computes the similarity between their multiset of observed values, but applying the instance matching heuristic, using the same similarity function and an instance matching, as explained before;
- uses the best result to decide if the pair of properties match

Section 4 further elaborates on this more sophisticated strategy of computing property matchings with the help of similarity functions, and introduces the notion of similarity model in detail.

4 Experiments

The experiments aimed at comparing the performance of the similarity functions and heuristics discussed in section 3, to calibrate the similarity models and estimate the performance of the best similarity model.

To accomplish these goals, we conducted a cross validation process using data about four classes of objects as the training and test corpora: *books*, *geographic places*, *music* and *video*. The data about books were obtained from the eBay [EBAY], Yahoo! [Yahoo!], Amazon [Amazon] and Barnes & Noble. The data about geographical places were obtained from two geographic gazetteers, GeoNames [GEONAMES] and the Alexandria Gazetteer [ADL]. The data about music and video were obtained from eBay, Yahoo! and Amazon. All data sources provide Web service access, except Barnes & Noble, in which case we developed an HTML parser to capture data from query results.

For each class, we first defined a bootstrap set of keywords which we used to query the databases. From the query results, we extracted the less frequent words. We then used these words 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 extensional schema matching technique using instance matching heuristic.

However, the music and video data exhibited a very low amount of instance matchings between different sources. We decided not to retrieve additional music and video data to induce experiments where there is potential loss of performance for the instance matching heuristic. By doing so, we achieve a more realistic calibration and performance estimation of the similarity model, since in general one should also face a scenario with an insufficient number of instance matchings.

The total number of records extracted from each source, by object class, is shown in Table 1 and the number of instance matchings between sources is presented in Table 2.

Table 1. Number of Records by Class and Source

	Class				
source	books	music	places	video	Total
Alexandria			19791		19791
Amazon	16410	7341		5760	29511
BarnesAndNoble	99791				99791
eBay	2264	564		245	3073
Geonames			3599		3599
Yahoo!	2172	817		621	3610
Total	120637	8722	23390	6626	159375

Table 2. Number of Instance Matchings between Sources

		Source					
Source		Alexan-dria	Amazon	Barnes	eBay	Geonames	Yahoo
books	Amazon	-	434	-	-	-	-
	Barnes	-	37	6	-	-	-
	eBay	-	736	336	8238	-	-
	Yahoo	-	398	17	386	-	6
music	Amazon	-	690	-		-	
	eBay	-	0	-	222	-	
	Yahoo	-	4	-	0	-	38
place	Alexandria	63856	-	-	-	-	-
	Geonames	4953	-	-	-	1936	-
video	Amazon	-	936	-	-	-	-
	eBay	-	0	-	420	-	-
	Yahoo	-	0	-	0	-	68

Since our primary goal was to test the property similarity functions and heuristics, we used simple instance matching techniques. Book instances match if they have the same ISBN, geographic places match if their centroids have approximately the same geographic coordinates and if their names are similar (the similarity in this case is the amount of equal words in their names). For music and video instances, we used the cosine similarity between the token sets of each instance, where the token set of an instance is obtained by tokenizing the values of the properties *category*, *name*, *description*, *artist* and *director*. The accuracy of this strategy was not systematically measured, but a manual inspection of the instance matchings obtained indicated that the strategy is indeed adequate. In addition, different similarity threshold were used to verify the impacts on the final result of the property matchings. We concluded that a threshold equal to 0.8 is precise enough for good property matchings.

Tables 3-6 show the RDF database schemas of the sources used in the experiments.

Table 3. RDF Database Schemes related to Books

Source	**Properties**
Amazon (amz-)	actor; artist; asin; author; brand; color; department; director; edition; format; index; isbn; label; listprice; manufacturer; model; operatingsystem; platform; productgroup; producttype; publisher; size; title; url
Barnes and Noble (bn-)	by; category; isbn-13; name; number-of-pages; pub-date; publ; sales-rank; subject
eBay (eby-)	item-id; category; title; sub-title; quantity; start-price; currency; location; postal-code; payment-methods; return-policy-item-must-be-returned-

	within; return-policy-refund-will-be-given-as; return-policy-return-policy-details; textbooks-education-author; textbooks-education-category; textbooks-education-edition-description; textbooks-education-format; textbooks-education-isbn-10; textbooks-education-isbn-13; textbooks-education-publication-year; textbooks-education-publisher; textbooks-education-sub-category; textbooks-education-title; textbooks-education-educational-level; textbooks-education-product-type; (*.. 70 other properties*)
Yahoo! (yho-)	department; description; id; merchant; name; price; pricefrom; priceto; summary; upc; url

Table 4. RDF Database Schemes related to Geographical Places

Source	Properties
Alexandria (adl_)	codes; relationships; classes; names; placestatus; boundingboy2; boundingboxx2; boundingboxy1; boundingboxx1; footprinty; footprintx; displayname; identifier
Geonames (gnm_)	elevation; alternatenames; population; adminname2; adminname1; admincode2; admincode1; fcodename; fclname; fcode; countryname; countrycode; lng; lat; name; id

Table 5. RDF Database Schemes related to Music

Source	Properties
Amazon (amz_)	actor; artist; asin; aspectratio; author; brand; color; director; edition; format; genre; index; isbn; label; listprice; manufacturer; model; operating-system; productgroup; producttype; publisher; size; style; title; url
eBay (eby_)	item-id; category; title; sub-title; quantity; start-price; currency; location; postal-code; payment-methods; return-policy-item-must-be-returned-within; return-policy-refund-will-be-given-as; return-policy-return-policy-details; return-policy_returns-accepted; music-cassettes_artist; music-cassettes_genre; music-cassettes_title; music-cds_artist; music-cds_genre; music-cds_title; music-other-formats_genre; music-records_genre; (*.. 43 other properties*)
Yahoo! (yho_)	department; description; id; merchant; name; price; pricefrom; priceto; summary; upc; url

Table 6. RDF Database Schemes related to Video

Source	Properties
Amazon (amz_)	actor; artist; asin; aspectratio; author; brand; color; cpuspeed; department; director; edition; format; genre; index; isbn; label; listprice; manufacturer; model; operating-system; platform; productgroup; producttype; publisher; size; style; title; url; warranty
eBay (eby_)	item-id; category; title; sub-title; quantity; start-price; currency; location; postal-code; payment-methods; return-policy-item-must-be-returned-within; return-policy-refund-will-be-given-as; return-policy-return-policy-details; return-policy_returns-accepted; vds_director; dvds_genre; dvds_title; movies-other-formats_genre; vhs_director; vhs_genre; (*.. 45 other properties*)
Yahoo! (yho_)	department; description; id; merchant; name; price; pricefrom; priceto; summary; upc; url

The experiment consisted of a 4-fold [Witten e Frank 2005] cross validation. The similarity models were trained with each combination of three classes and tested with the class not used for training. For each test class, the performance of the model was computed.

We considered as the best model that with the highest average performance. The performance measure is defined as [Manning e Schütze 2002]:

$$fMeasure = \frac{1}{(2 * precision)^{-1} + (2 * recall)^{-1}}$$

Note that, since we are trying to obtain property matchings, *recall* measures the set of pairs of properties from the different schemes that were found to match, as compared to the set of pairs of properties from the different schemes that were defined in the reference property matching. *Precision* and *fMeasure* are likewise defined.

Before analysing the training algorithm, a brief definition of the notions of similarity model and model parameters is needed. We define a *similarity model* as a tuple of the form

$$M = (\sigma, \textit{domain}, \textit{instance}, \textit{multiset}, \textit{type compatibility})$$

where

- σ is one of the similarity functions defined in Section 3.1,
- domain, instance, multiset and type compatibility are Boolean values indicating
  - domain = True : the model applies the similarity function to sets of observed domains
  - instance = True : the model uses the instance matching heuristic
  - multiset = True : the model uses the multiset heuristic
  - type co*mpatibility = True* : the model uses the type compatibility heuristic.

Note that, when *domain=instance=True*, the model actually considers the best result from applying the similarity function to sets of observed domains and from applying the instance matching heuristic. Additionally, when *multiset=true,* the model applied multiset heuristic to the observed values and to the sets using the instance matching heuristic, likewise *type compatibility=true* is applied to both sets.

Each model has to be calibrated by adjusting the thresholds s_d, s_c and, in the case of models using the customized similarity function, by adjusting also the parameters θ, α and β in order to induce the best similarity matchings. We call this set of constants the *parameters* of the model.

Now, the training algorithm can be written as follows:

```
1. for each set of three training classes
   1.1. for each model
        1.1.1. for each combination of parameters $(\theta, \alpha, \beta, s_d, s_c)$
               1.1.1.1. for each class in the set
                        1.1.1.1.1. generate the property matching
                        1.1.1.1.2. compare the results with the reference matching
                        1.1.1.1.3. calculate fMeasure
               1.1.1.2. calculate the average fMeasure
   1.2. choose the parameter set with the best average performance, and apply it
        (steps 1.1.1.1.1 to 1.1.1.1.3) to the class which is not in the set
2. calculate the average fMeasure and choose the model with the highest average
```

We tested models with parameters $\theta=1$, $\alpha=\beta\in[0.5,5.0]$, $s_t=0.8$ (only for music and video), and $s_d=s_c\in[0.0,1.0]$. The parameter intervals were chosen based on preliminary evaluations of the formula in (8) of Section 3.1, with different parameter values, which indicated intervals that would be good guesses of the optimal region.

The reference property matchings were manually defined by inspecting instances and by an intuitive syntactical analysis of property names.

Tables 7-10 summarize the reference property matchings, where each row contains a group of matching properties. Note that each line of Table 7 indicates that several properties from eBay match a single property from Amazon and from Barnes and Noble. Hence, the reference matching is not one-to-one.

Table 7. Books – Reference Property Matchings

Group	Properties
1	amz-title; bn-name; eby-audiobooks-title; eby-children-s-books-title; eby-fiction-books-title; eby-nonfiction-books-title; eby-textbooks-education-title; eby-title; yho-name;
2	amz-author; bn-by; eby-audiobooks-author; eby-children-s-books-author; eby-fiction-books-author; eby-nonfiction-books-author; eby-textbooks-education-author; yho-description;
3	amz-index; amz-productgroup; bn-category; eby-category; yho-department;
4	amz-asin; amz-isbn; eby-audiobooks-isbn-10; eby-children-s-books-isbn-10; eby-fiction-books-isbn-10; eby-nonfiction-books-isbn-10; eby-textbooks-education-isbn-10; yho-upc;
5	bn-isbn-13; eby-audiobooks-isbn-13; eby-children-s-books-isbn-13; eby-fiction-books-isbn-13; eby-nonfiction-books-isbn-13; eby-textbooks-education-isbn-13;
6	amz-label; amz-manufacturer; amz-publisher; bn-publ; eby-audiobooks-publisher; eby-children-s-books-publisher; eby-fiction-books-publisher; eby-nonfiction-books-publisher; eby-textbooks-education-publisher;
7	bn-pub-date; eby-children-s-books-publication-year; eby-fiction-books-publication-year; eby-magazine-back-issues-publication-year; eby-nonfiction-books-publication-year; eby-textbooks-education-publication-year;
8	amz-edition; eby-children-s-books-edition; eby-fiction-books-edition; eby-nonfiction-books-edition; eby-textbooks-education-edition;
	(…13 other equivalence groups)

Table 8. Places – Reference Property Matchings

Group	Properties
1	adl_boundingboxx1; adl_boundingboxx2
2	adl_boundingboxy1; adl_boundingboy2
3	adl_displayname; adl_names; gnm_alternatenames; gnm_name
4	adl_footprintx; gnm_lng
5	adl_footprinty; gnm_lat
6	adl_classes; gnm_fclname; gnm_fcodename
7	gnm_admincode1; gnm_admincode2
8	gnm_adminname1; gnm_adminname2

Table 9. Music – Reference Property Matchings

Group	Properties
1	amz_artist; eby_music-cassettes_artist; eby_music-cds_artist
2	amz_title; eby_music-cassettes_title; eby_music-cds_title; eby_title; yho_description; yho_name
3	amz_index; amz_productgroup; yho_department
4	amz_isbn; eby_music-cassettes_upc; eby_music-cds_upc; yho_upc
5	amz_url; yho_url
6	amz_listprice; eby_start-price; yho_price; yho_pricefrom
7	eby_music-cassettes_format; eby_music-cds_format; eby_music-other-formats_format
10	(…8 other equivalence groups)

Table 10. Video – Reference Property Matchings

Group	Properties
1	amz_director; eby_dvds_director; eby_vhs_director
2	amz_title; eby_dvds_title; eby_title; yho_description; yho_name

3	amz_index; amz_productgroup; eby_category; yho_department;
4	amz_isbn; eby_dvds_upc; eby_vhs_upc; yho_upc
5	amz_url; yho_url
6	amz_listprice; eby_start-price; yho_price
7	amz_format; eby_dvds_format; eby_movies-other-formats_format; eby_vhs_format
	(…10 other equivalence groups)

Tables 11-12 show the best 10 models and their calibrations. The *fMeasure* shown in Table 11 is the average *fMeasure* of the four test classes in the cross validation process. As Table 11 shows, the similarity function based on the contrast model is 3% better than that based on information theory, presented in [Lin 1998]. The experiment based on cosine similarity appears in the third position, and the model proposed in [Brauner, Gazola e Casanova 2008] did not rank well.

Table 11. Best 10 Models of Similarity

similarity function	set of values		heuristics		*fMeasure*
	instance	attribute	multiset	type comp.	
contrast model	-	used	-	used	62%
contrast model	used	used	-	used	60%
information theory	-	used	-	used	59%
information theory	used	used	-	used	58%
Cosine	used	used	used	used	57%
Cosine	-	used	used	used	57%
contrast model	used	used	used	used	53%
information theory	used	used	used	used	52%
contrast model	-	used	used	used	51%
information theory	-	used	used	used	50%

Table 11. Calibration of the Best 10 Models of Similarity

similarity function	set of values		heuristics		calibration	
	instance	attribute	multiset	type comp.	α,β	*s*
contrast model	-	used	-	used	3,5	0,10
contrast model	used	used	-	used	3,5	0,10
information theory	-	used	-	used	-	0,10
information theory	used	used	-	used	-	0,15
cosine	used	used	used	used	-	0,15
cosine	-	used	used	used	-	0,15
contrast model	used	used	used	used	2,5	0,10
information theory	used	used	used	used	-	0,05
contrast model	-	used	used	used	2,5	0,10
information theory	-	used	used	used	-	0,05

Contrary to our expectations, the best model did not use the instance matching heuristic, which is partly explained by the low number of instance matchings for the music and video data. The experiments also showed that the best model for books – which had 10.594 instance matchings – had a performance of 63%, using the instance matching heuristic and type compatibility. However, using both sets of values (instance values and attribute values) the similarity model does not lose much performance, but it tends to be more general, in the sense that the use of attribute values compensates the lack of instance matchings in the data sources.

Fig. 4 shows the performance measures of the best model applied to the four classes of objects. The average group is the average of the performance measures of all four classes. Examining this chart, we can formulate the hypothesis that it is reasonable to assume that the parameters of the best model may apply to a wider range of application domains since the performance for all data classes is very similar. Additional tests should be made to verify this behavior.

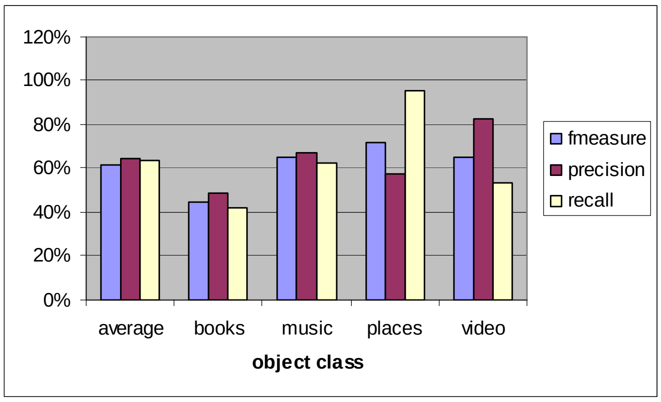

Fig. 4 Performance of the best model on each data class. The average group is the average performance measures of all four data classes.

The best model had an average *fMeasure* of 62%, with an average precision of 64%, and an average recall of 63%. The model used the domain similarity and type compatibility heuristics. The next models successively relax each of the three heuristics and vary the parameter values. As Table 11 shows, they get gradually worse. However, the second best model, shown in Fig. 5 – with *fMeasure*=62%, *precision*=61%, and *recall*=63% – is a good alternative since it loses only 2% in the cross validation test, maintains the average performance on the four classes and uses both similarities: instance matchings and domain.

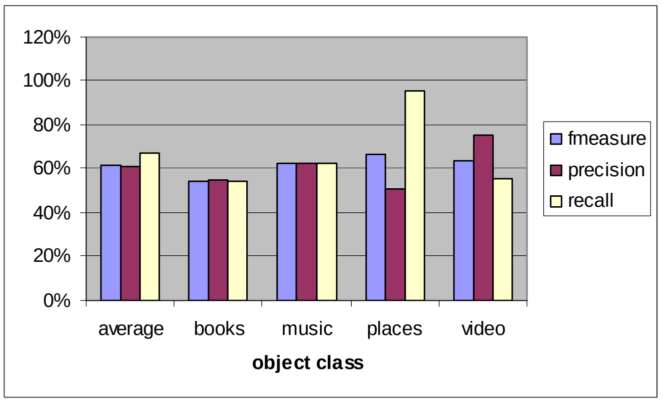

Fig. 5 Performance of the second best model on each data class. The average group is the average performance measures of all four data classes.

Fig. 6 shows the graph of the average performance measures of the best model as a function of the threshold level. Observe that the precision of the model increases with the threshold level, but the recall decreases. The decreasing of the precision with the precision above 0.15 is due to the normalization we did in equation (5) in section 3.1 which caused the customized contrast model do not exceed a certain value.

The same graph for each class shows a more abruptly decreasing. It is consistent since the value |C|, which is used to normalize the similarity function, is calculated for each class. This behaviour is useful to control the response to users. If so requested, the system may decrease the threshold level and may thereby return more answers, although with lower precision.

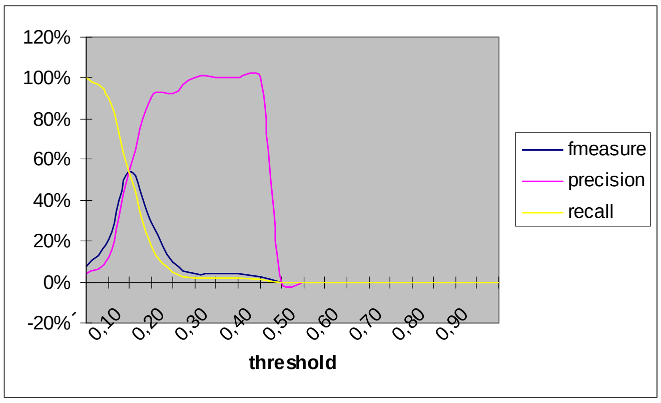

Fig. 6 Average performance measures of the best model by threshold

5 Related work

Rahm and Bernstein (2001) is an early survey of schema matching techniques. Euzenat and Shvaiko (2007) contain an account of ontology matching techniques. Following their classification, the technique described in this paper is classified as extensional and based on data analysis and statistics. Bernstein and Melnik (2007) list the requirements for model management systems that support schema mappings, to which the work reported in this paper contributes.

Bilke and Naumann (2005) describe an extensional technique based on similarity algorithms. Brauner et al. (2006) adopt the same idea to match two thesauri. Wang et al. (2004) describe a technique based on query probing to match Web databases, which relies on human intervention to select a set of typical instances used in the probing. Brauner et al. (2007) apply this idea to match geographical database Web services. Brauner et al. (2008) describe a matching algorithm based on measuring the similarity between the property domains of distinct Web databases. Madhavan et al. (2005) propose the use of a set of schemas and mappings to help the schema matching algorithms. The authors use predictor algorithms that measure the similarity between schema elements, adopted in the PayGo architecture [Madhavan et al. 2007].

Contrasting with [Wang et al. 2004], we work with Web services that encapsulate databases. This assumption supports the heuristics introduced in section 3.2. Also, differently from [Wang et al. 2004] and [Brauner et al. 2007], and we avoid the use of a global schema and a set of global instances, which are sometimes hard to define. The work reported here also contrasts with [Bilke e Naumann 2005], [Madhavan et al. 2005] and [Brauner, Gazola e Casanova 2008] in that the property matching functions adopted are based on customized Tversky contrast models, which were not explored before for schema matching, and may result in more general many-to-many property matchings. Section 4 shows that this level of generality may sometimes be required. The results in Section 4 also explore how the precision of the property matchings is influenced by heuristics that take into account property types, as well as instance matchings. The results favor property matchings induced by customized Tversky contrast models, as compared to other well-know similarity functions.

6 Conclusions

The investigation reported in this paper contributes to the definition of schema matching strategies classified as extensional and based on data analysis and statistics [Euzenat e Shvaiko 2007]. To provide the foundations of the discussion, we first defined the concepts of property matching and instance matching, and discussed how to use similarity functions to induce matchings. Then, we introduced property matchings induced by customized Tversky contrast models. Finally, we described experimental results evaluating the precision of the property matchings induced by customized Tversky contrast models, and measuring the influence of the proposed heuristics.

References

ADL, *Alexandria Digital Library Gazetteer*. 1999. Santa Barbara CA: Map and Imagery Lab, Davidson Library, University of California, Santa Barbara. Copyright UC Regents. Disponvel em: http://www.alexandria.ucsb.edu/gazetteer

AMAZON, *Amazon Webservice*. Disponvel em: http://developer.amazonwebservices.com/connec

BERNSTEIN, P.; MELNIK, S. Model management 2.0: manipulating richer mappings. In: *Proc. 2007 ACM SIGMOD Int´l. Conf. on Management of Data*. New York, NY, USA: [s.n.], 2007. p. 1–12.

BILKE, A.; NAUMANN, F. "Schema matching using duplicates". In: *Proc. 21st Int´l. Conf. on Data Engineering*. Tokyo, Japan: [s.n.], 2005. p. 69–80. ISBN 0769522858. ISSN 1084-4627.

BRAUNER, D. F.; CASANOVA, M. A.; MILIDIÚ, R. L. Mediation as recommendation: an approach to the design of mediators for object catalogs. In: *Proc. 5th Int. Conf. on Ontologies, DataBases, and Applications of Semantics*. Montpellier, France: LNCS, 2006. v. 4277, p. 46.

BRAUNER, D. F.; GAZOLA, A.; CASANOVA, M. A. Adaptative matching of database Web services export schemas. In: *10th Int'l. Conf. on Enterprise Information Systems*. Barcelona, Spain: [s.n.], 2008.

BRAUNER, D. F. et al. An instance-based approach for matching export schemas of geographical database web services. In: VINHAS, L.; COSTA, A. C. da R. (Ed.). *GeoInfo*. INPE, 2007. p. 109–120. ISBN 9788517000362.

BREITMAN, K.; CASANOVA, M.; TRUSZKOWSKI, W. *Semantic web: concepts, technologies, and applications*. [S.l.]: Springer, London, 2007.

COVER, T.; THOMAS, J. *Elements of Information Theory*. [S.l.]: Wiley-Interscience New York, 1991.

EBAY, eBay Trading API. Disponvel em: http://developer.ebay.com.

EIDENBERGER, H. Evaluation and analysis of similarity measures for content-based visual information retrieval. *Multimedia Systems*, Springer, v. 12, n. 2, p. 71–87, 2006.

EIDENBERGER, H.; BREITENEDER, C. Visual similarity measurement with the feature contrast model. In: *Proc. of SPIE - Storage and Retrieval for Media Databases Conf.* Santa Clara: SPIE, 2002. v. 5021, p. 64–76.

EUZENAT, J.; SHVAIKO, P. *Ontology matching*. New York: Springer-Verlag, 2007.

FRAKES, W.; BAEZA-YATES, R. *Information retrieval: data structure and algorithms*. [S.l.]: Prentice Hall, 1992.

GEONAMES, *GeoNames Geographical Database*. Disponvel em: http://www.geonames. org /export.

HINDLE, D. Noun classification from predicate-argument structures. In: *Proc. of the 28th annual meeting on Association for Computational Linguistics*. Morristown, NJ, USA: ACM Press, 1990. p. 268–275.

LEE, J. Information retrieval based on conceptual distance in Is-a hierarchies. *Journal of Documentation*, MCB UP Ltd, v. 49, n. 2, p. 188–207, 1993.

LIN, D. An information-theoretic definition of similarity. In: *Proc. of the 15th Int´l. Conf. on Machine Learning*. Madison, WI, USA: [s.n.], 1998. p. 296–304.

MADHAVAN, J. et al. Web-scale data integration: You can afford to pay as you go. In: *Proc. of 3rd Biennial Conf. on Innovative Data Systems Research*. Asilomar, California: www.crdrdb.org, 2007. p. 342–350.

MADHAVAN, J. et al. Corpus-based schema matching. In: *Proc. 21st Int´l. Conf. on Data Engineering*. Tokyo, Japan: [s.n.], 2005. p. 57–68. ISSN 1084-4627.

MANNING, C. D.; SCHÜTZE, H. *Foundations of Statistical Natural Language Processing*. [S.l.]: The MIT Press, 2002.

RAHM, E.; BERNSTEIN, P. A survey of approaches to automatic schema matching. *The VLDB Journal*, Springer, v. 10, n. 4, p. 334–350, 2001.

RESNIK, P. Using information content to evaluate semantic similarity in a taxonomy. In: *Proc. of the 14th Int´l. Joint Conf. on Artificial Intelligence*. Montreal, Canadá: [s.n.], 1995. p. 448–453.

TANG, H. et al. Similarity measures for satellite images with heterogeneous contents. In: MAITRE, H. (Ed.). *Proc. Urban Remote Sensing Joint Event*. [S.l.: s.n.], 2007. p. 1–9.

TVERSKY, A. Features of similarity. *Psychological Review*, v. 84, n. 4, p. 327–352, 1977.

WANG, J. et al. Instance-based schema matching for web databases by domain-specific query probing. In: *Proc. of the 13th Int´l. Conf. on Very Large Data Bases*. Toronto, Canada: [s.n.], 2004. p. 408–419. ISBN 0120884690.

WITTEN, I.; FRANK, E. *Data Mining: Practical Machine Learning Tools and Techniques*. [S.l.]: Morgan Kaufmann, 2005.

YAHOO, *Yahoo! Shopping Web Services*. Disponvel em: http://developer.yahoo. com/shopping.