Obtaining Optimum Ontology Matching Functions by Using Heuristic Approaches

Jorge Martinez-Gil
University of Málaga, Department of Languages and Computing Sciences
Boulevard Louis Pasteur s/n 29071 Málaga (Spain)
E-mail: jorgemar@lcc.uma.es

Abstract. Nowadays there are a lot of techniques and tools for addressing the ontology matching problem, however, the complex nature of this problem causes existing solutions to be unsatisfactory. This work intends to shed some light on a more flexible way of matching ontologies: ontology meta-matching. In this sense, we think that an exhaustive study of the problem and a examination of the existing leading-edge solutions will help developers build more accurate and dynamic systems for selecting the appropiate algorithms, weights and thresholds in each ontology alignment scenario.

Keywords: meta-matching, ontologies, semantic web

1. Introduction

The Semantic Web is a promising paradigm for the Web in which the semantics of information is defined, making it possible for the Web to understand and satisfy all of the requests from people and machines to use the web resources. Therefore, most of the authors consider it as a vision of the Web as an universal medium for data, information, and knowledge exchange (1).

In relation to knowledge, it is very important the notion of ontology as a form of representation about a particular universe of discourse or some part of it. Ontology matching is a key aspect in order to the knowledge exchange in this extension of the Web may be real; it allows organizations to model their own knowledge without having to stick to a specific standard. In fact, there are two good reasons why most organizations are not interested in working with a standard for modelling their own knowledge: (a) it is very difficult or expensive for many organizations to reach a agreement about a common standard, and (b) these standards do not often fit to the specific needs of the all participants in the standarization process.

Therefore, the old problem of matching schemas has evolved into an analog problem, although it is a little more complex, because now we handle knowledge. The task of finding correspondences between ontologies is called ontology matching and the output of this task is called ontology alignment (2). In fact, obtaining satisfactory ontology alignments is a key aspect for such fields as:

- Semantic integration (3). It is the process of combining metadata residing at different sources and providing the user with a unified view of these data. This kind of integration should be done automatically, because manual integration is not viable, at least, for large volumes of information (4).
- Ontology mapping (5). It is used for querying different ontologies. An ontology mapping is a function between ontologies. The original ontologies are not changed, but the additional mapping axioms describe how to express concepts, relations, or instances in terms of the second ontology. They are stored separately from the ontologies themselves. A typical use case for mapping is a query in one ontology representation, which is then rewritten and handed on to another ontology. The answers are then mapped back again. Whereas alignment merely identifies the relation between ontologies, mappings focus on the representation and the execution of the relations for a certain task.
- The Web Services industry, where there is tendency to discover and compose Semantic Web Services (SWS) (6) in a completely unsupervised manner. Originally SWS alignment was based on exact string

matching of parameters, but nowadays researchers deal with issues of heterogeneous and constrained matching[1].
– Similarity-based retrieval (7). Semantic similarity measures play an important role in information retrieval by providing means to improve recall and precision. They are used in various application domains ranging from product comparison to job recruitment.

On the other hand, although automatic matching is perhaps the most appropriate way to align ontologies, it has the disadvantage that "finding good similarity functions is, data-, context-, and sometimes even user-dependent, and needs to be reconsidered every time new data or a new task is inspected" (8). Moreover, when dealing with natural language often it leads a significant error rate, so researchers try to find *customized similarity functions* (CSF) in order to obtain the best alignment for each situation.

On the other hand, functions for calculating alignments can be divided into similarity measures and distance measures.

– A similarity measure is a function that associates a numeric value with a pair of objects, with the idea that a higher value indicates greater similarity.
– A distance measure is a function that associates a non-negative numeric value with a pair of objects, with the idea that a short distance means greater similarity. Distance measures usually satisfy the mathematical axioms of a metric.

Mathematical laws used to describe behaviour in one domain are not always appropriate in others domains, and there are long-standing psychological objections to the axioms used to define a distance metric. For example, a metric will always give the same distance from a to b as from b to a, but in practice we are more likely to say that a child resembles their parent than to say that a parent resembles their child. Similarity measures, however, give us an idea about the probability of compared objects being the same, but without falling into the psychological objections of a metric. So from our point of view, working with similarity measures is more appropiate for detecting correspondences between different ontologies belonging to a same domain. In this sense, Ontology Meta-Matching could be considered as a technique that automatically selects the appropiate similarity measures and its associated weights in case that similarity measures need to be composed. The main contributions of this work are:

– An introduction to the problem of Ontology Meta-Matching.
– An original structural similarity measure for align ontologies (SIFO).
– An original linguistic similarity measure which use search engines on the Internet to align ontologies.
– An original statistical similarity measure which use textual analysis for aligning ontologies.
– An original greedy algorithm for solving the Meta-Matching problem automatically and efficiently (MaSiMe).
– An original genetic algorithm for optimizing the solutions to that problem (GOAL).
– An empirical evaluation of the proposed algorithms and a discussion about their convenience.
– An exhaustive study of the future research lines related to this field.

The remainder of this article is organized as follows. Section 2 describes the problem statement related to the Ontology Meta-Matching problem. Section 3 describes the preliminary definitions and properties that are neccesary for our proposal. Section 4 describes the development of an algorithm for computing a structural similarity measure. Section 5 describes a linguistic similarity measure which uses Internet as source of background knowledge. Section 6 discusses the implementation of a statistical method to align ontologies. Section discusses a greedy strategy and a way to effectively compute it. Section 7 describes a genetic strategy and a way to effectively compute it. Section 8 shows the empirical data that we have obtained from some experiments, including results obtained from a standard benchmark for ontology matching. Section 9 includes the related work section which compares our work with other approaches. And finally, in Section 10 the conclusions are discussed and the future research lines presented.

[1] http://insel.flp.cs.tu-berlin.de/wsc06/

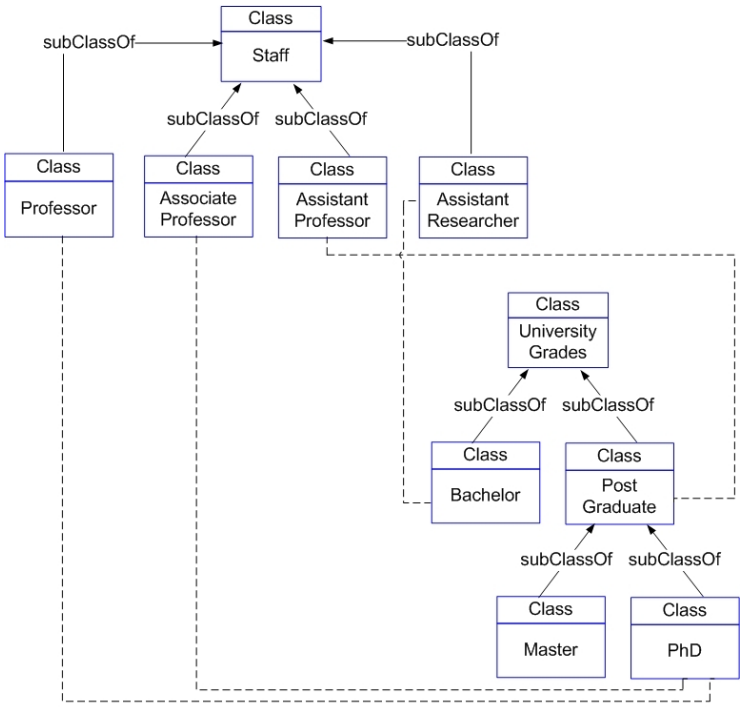

Fig. 1. Example of an user-dependent alignment

2. Problem Statement

The process of aligning ontologies can be expressed as a function f where given a pair of ontologies o and o', an input alignment A, a set of parameters p and a set of resources r, returns an alignment A':

$$A' = f(o, o', A, p, r)$$

Where A' is a set of mappings. A mapping[2] is an expression that can be written in the form (id, e, e', n, R). Where id is an unique identifier for the mapping, e and e' are entities belonging to different ontologies, R is the relation of correspondence and n is a real number between 0 and 1, which represents the mathematical probability that R may be true (9). The entities than can be related are the concepts, roles, rules and, even axioms of the ontologies.

However, experience tells us that finding f is far from being trivial. As we commented earlier, the heterogeneity and ambiguity of data description makes unavoidable that optimal mappings for many pairs of entities will be considered as "best mappings" by none of the existing ontology matching algorithms which are usually called matchers. For this reason, it is neccesary to compose these matchers. Figure 1 shows an example of an user-dependent alignment between ontologies, because this alignment is not valid for all the countries in the world.

Composite matchers are an aggregation of simple matching algorithms. They exploit a wide range of information, such as ontology characteristics (i.e. metadata, such as element names, data types, and structural properties), characteristics of data instances, as well as background knowledge from dictionaries and thesauri.

1. **String normalization.** This consists of methods such as removing unnecessary words or symbols from the entity names. Moreover, they can be used for detecting plural nouns or to take into account common prefixes or suffixes as well as other natural language features.

[2]Output tuples from an alignment are called mappings. But using ontology alignments for query purposes is called ontology mapping.

2. **String similarity.** Text similarity is a string based method for identifying similar entity names. For example, it may be used to identify identical concepts of two ontologies if they have a similar name. The reader can see (10) for more details about this algorithms.
3. **Data Type Comparison.** These methods compare the data type of the ontology elements. Similar concept attributes are logically expected to have the same data type.
4. **Linguistic methods.** This consists of the inclusion of linguistic resources such as lexicons and thesauri to identify possible similarities. The most popular linguistic method is to use WordNet (11) to identify some kinds of relationships between entities.
5. **Inheritance analysis.** Theses kinds of methods take into account the inheritance between concepts to identify relationships. The most popular method is the *is-a* analysis that tries to identify subsumptions between concepts.
6. **Data analysis.** These kinds of methods are based on the rule: If two concepts have the same instances, they will probably be similar. Sometimes, it is possible to identify the meaning of an upper level entity by looking at a lower level entity. For example, if instances contain a string such as *years old*, it probably belongs to an attribute called *age*.
7. **Graph-Mapping.** This consists in identifying similar graph structures in two ontologies. These methods use known graph algorithms to do so. Most of times this involves computing and comparing paths, adjacent nodes and taxonomy leaves.
8. **Statistical analysis.** It consists of the extraction of keywords and textual descriptions for detecting the meaning of the entities in relation to other entities.
9. **Taxonomy analysis.** It tries to identify similar concepts by looking at their related concepts. The main idea is that two concepts belonging to different ontologies have a certain degree of probability of being similar if they have the same neighbours.
10. **Semantic methods** According to (2), semantic algorithms handle the input based on its semantic interpretation. One supposes that if two entities are the same, then they share the same interpretations. Thus, they are well grounded deductive methods. Most outstanding approaches are propositional satisfiability and description logics reasoning techniques.

However, choosing from among this variety of algorithms is far from being a trivial task. Firstly, more and more are constantly being developed, and this diversity by itself complicates the choice of the most appropriate tool for a given application domain. Secondly, as one would expect, recent empirical analysis shows that there is no (and may never be) single dominant matcher that performs best, regardless of the data model and application domain (12). In fact, due to effectively unlimited heterogeneity and the ambiguity of data description, it seems unavoidable that optimal mappings for many pairs of correspondences will be considered as *best mappings* by none of the existing matchers. For this reason, it is neccesary to use composite matchers.

The main idea of composite matchers is to combine similarity values predicted by multiple similarity measures to determine correspondences between ontology elements. The most outstanding proposals in this way are COMA (13), an extension of COMA (14), QuickMig (15), FOAM (16), iMAP (17) and OntoBuilder (18), Cupid (19), CMC (20), and MAFRA (21) but they use weights determined by an expert. Meta-Matching does not use weights from an expert, but selects those that would solve the problem according to a training benchmark, thus a set of ontologies that have been previously aligned by an expert.

3. Technical Background

Definition 1 (Similarity Measure). *A similarity measure sm is a function* $sm : \mu_1 \times \mu_2 \mapsto R$ *that associates the similarity of two input solution mappings* μ_1 *and* μ_2 *to a similarity score* $sc \in \Re$ *in the range [0, 1].* This definition has been taken from (12)

A similarity score of 0 stands for complete inequality and 1 for equality of the input solution mappings μ_1 and μ_2.

Definition 2 (Customizable Similarity Measure). *A Customizable similarity measure is a similarity measure that can be parametrized.* Example 1 shows a function of this type.

Example (Weighted Similarity Measure). Let $\vec{A}$ be a set of similarity measures and $\vec{w}$ a weight vector and let O_1, O_2 be two input ontologies, then we can define wsm as a weighted similarity measures in the following form:

$$wsm(O_1, O_2) = x \in [0,1] \in \Re \rightarrow \exists \left\langle \vec{A}, \vec{w} \right\rangle, x = max(\sum_{i=1}^{i=n} A_i \cdot w_i)$$
$$\textit{with the following restriction } \sum_{i=1}^{i=n} w_i \leq \kappa$$

But from the engineering point of view, this function leads to an optimization problem for calculating the weight vector, because the number of candidates from the solution space (in this case an arbitrary continous interval) is infinte. For this reason, a brute force strategy would clearly be inefficient. It is neccesary to look for better computational mechanisms that allow the problem of computing tuneable measures to be solved more efficiently.

Definition 3 (Ontology Matching). *An ontology matching om is a function $om : O_1 \times O_2 \stackrel{sm}{\rightarrow} A$ that associates two input ontologies O_1 and O_2 to an alignment A using a similarity measure (or a customizable similarity measure).*

Definition 4 (Ontology Alignment). *An ontology alignment oa is a set $\{t, MD\}$. t is a set of tuples in the form $\{(id, e, e', n, R)\}$. Where id is a unique identifier, e and e' are entities belonging to two different ontologies, R is the relation of correspondence between these entities and n is a real number between 0 and 1 that representing the mathematical probability that R may be true. The entities than can be related are the concepts, roles, rules and, even axioms of the ontologies. On the other hand, MD is some metadata related to the matching process for statistical purposes.*

Definition 5 (Alignment Evaluation). *An alignment evaluation ae is a function $ae : A \times A_R \mapsto precision \in \Re \in [0,1] \times recall \in \Re \in [0,1]$ that associates an alignment A and a reference alignment A_R to two real numbers stating the precision, recall of A in relation to A_R.*

Definition 6 (Meta-Matching Function). *A Meta-Matching Function mmf is a function $mmf : SC \mapsto \Re$ that defines how previously calculated similarity score $sc_i \in SC$. The result is an optimized similarity score $sc_o \in \Re$.* We call optimized similarity score to the best possible similarity score.

4. Structural Matching

This work discusses the design and development of a taxonomy-based algorithm to extract some valuable information from a kind of ontology entities: concepts. The algorithm can help us to decide if two concepts are the same but from the point of view of their location in an ontology. This kind of information can be useful in ontology alignment scenarios.

Definition 7. *Taxon is a name designating a class or group of classes. A taxon is assigned a rank and can be placed at a particular level in a systematic hierarchy reflecting relationships.*

Before to designing the algorithm, it is also necessary to define a data structure (DS) to store the data calculated by the algorithm. The data structure is a simple linked list with six records in each node: *depth, children, brothers, brothers_left, same_level and name*. Table 1 tells us the data type and a brief descrip-

Attribute	Type	Description
depth	*integer*	*Level of the current taxon (begins with 0).*
children	*integer*	*Number of children of the current taxon*
brothers	*integer*	*Number of brother taxons*
brothers_left	*integer*	*Number of brother taxons that are above this*
same_level	*integer*	*Number of taxons with the same depth*
name	*string*	*ID of the taxon*

Table 1
A node of the linked list which stores the information

```
entry procedure ont2tax (OntoClass cls, List occurs, int depth)
 begin
   storingInTax(cls, depth) ; Step 2
   if (cls.canAs( OntClass.class ) AND (NOT occurs.contains( cls )))
     while iterator = cls.SubClasses do
       OntClass sub := (OntClass) iterator.next
       occurs.add(cls)
       ont2tax (sub, occurs, depth + 1)
       occurs.remove(cls)
     end while
   end if
 end
```

Fig. 2. Procedure for converting an ontology into a taxonomy

tion of each of these records.

4.1. The Structural Algorithm

The algorithm that we propose is divided into three high level steps:

1. To convert the ontology into a taxonomy (See Fig. 2).
2. To store the taxonomy in some parts of a special data structure (See Fig. 3).
3. To order and fill the data structure with complementary calculations (See Fig. 4).

Finally, it is necessary to call the algorithm (See Fig. 5). The philosophy of the SIFO algorithm consists of detecting the changes in the depths of each taxon in the taxonomy. In this way, it is possible to count the different kinds of neighbours that a concept may have.

4.1.1. First Step. Converting an ontology into a taxonomy.

This is the first step which consists of converting the ontology into a taxonomy which will allow us to compute more easily the data related to the neighborhood of each concept into the ontology. The procedure is inspired by one offered by the Jena API to visit all the concepts in an ontology.

4.1.2. Second Step. Storing the taxonomy in the data structure.

The storingInTax method has the following interface: *storingInTax (int depth, int children, int brothers, int brotherontheleft, string name)* where each argument is the value for a new entry in the DS. However, in this second step, we only know the depth (number of indents for the taxon) and the name of each concept, so we can only partially fill the data structure, thus, we can only invoke the procedure with the arguments depth and concept name.

```
entry procedure storingInTax(OntoClass cls, int depth)
 begin
    Element e:= new Element (depth, 0, 0, 0, 0, cls.getName)
    DS.add (e)
 end
```

Fig. 3. Procedure for partially storing the taxonomy

Concept	*Depth*	*Children*	*Brothers*	*Left brothers*	*Same level*
SLR	0	0	5	0	5
$Money$	0	0	5	1	5
$BodyWithNonAd.$	0	0	5	2	5
$Range$	0	0	5	3	5
$Window$	0	4	5	4	5
$PurchaseableItem$	1	2	3	0	4
$Camera$	2	0	1	0	1
$Digital$	2	0	1	1	1
$Large-Format$	1	0	3	1	4
$Lens$	1	0	3	2	4
$Body$	1	0	3	3	4
$Viewer$	0	1	5	5	5
$HQ-Viewer$	1	0	0	0	4

Table 2

Data structure obtained from the taxonomy of Fig.6

4.1.3. Third Step. Ordering and filling the data structure.

With data stored in the DS, we can now detect the changes in the depth of the entries in the taxonomy to compute the number of children, brothers and, so on. It is necessary to take into account the following details:

- All taxons with the same depth are same level taxons.
- A chain of brothers is a chain of taxons at the same level.
- A change to an outer taxon breaks a chain of brothers.
- All brothers with a previous position are on the left.
- Given a taxon, if the following concept has an inner depth, it is a child.
- A chain of children can only be broken by a change to an outer taxon.
- An inner taxon (grandson taxon) does not break a chain of children.

Calling to the algorithm. Now, it is necessary to invoke the SIFO algorithm. At this point it is necessary to define the ontology model (for example OWL) and to locate the classes without ancestors, in order to begin to visit all the concepts. Note that the ArrayList is necessary to store the visited concepts. Fig. 5 shows the related portion of pseudocode.

4.1.4. A trace to the algorithm with a small example

We have a choosen a small ontology from a camera to see a trace of the algorithm. In Fig. 6., we can see a graphical representation of the ontology, in Fig. 7 we can see a taxonomic representation of such ontology, we have obtained this taxonomy as result of applying the step 1 of SIFO. Finally, in Table 2 we can see the completed data structure that has been obtained from steps 2 and 3 of SIFO.

4.2. Results

The purpose of this section is not only to show numeric results about a Java implementation of SIFO but, to show the relative ease with which a new ontology and taxonomy matchers can be developed, based

```
entry procedure fillin (void)
 var children, brothers, brothers_left:  integer
 var same_level, i, j, k, t: integer
 var flag: boolean
 begin
   for i := 0 to DS.size
      children, brothers, brothers_left := 0
      flag := false
      for j := 0 to DS.size
         if (j < i)
            if (DS[i].depth = DS[j].depth)
               brothers++
               brothers_left++
            end if
            if (DS[i].depth < DS[j].depth)
               brothers := 0
               brothers_left := 0
            end if
         end if
         if (j > i)
            if (DS[i].depth = DS[j].depth)
               brothers++
            end if
            ; detect and end-of-children
            if (DS[i].depth < DS[j].depth)
               break
            end if
         end if
         ; code for counting children
         if ((j = i+1) AND (DS[i].depth = DS[j].depth - 1) AND (NOT flag))
            for k := j to DS[j].depth < DS[k].depth
               if (DS[j].depth = DS[k].depth)
                  child++
                  flag := true
               end if
            end for
         end if
      end for
      for t := 0 to DS.size
         if (NOT t=i) AND (DS[i].depth = DS[t].depth)
            same_level++
         end if
      end for
      DS[i].addNumChildren (children)
      DS[i].addNumBrothers (brothers)
      DS[i].addNumBrothersOnTheLeft (brother_left)
      DS[i].addNumSameLevel (same_level)
   end for
end
```

Fig. 4. Procedure for reordering and filling the data structure

```
OntoModel m := createOntologyModel
Iterator i := m.listHierarchyRootClasses()
while i.hasNext() do
   onto2tax((OntClass) i.next(), new ArrayList(), 0 )
end while
fillin ()
```

Fig. 5. Calling to SIFO

```
camera:SLR
camera:Money
camera:BodyWithNonAdjustableShutterSpeed
camera:Range
camera:Window
   camera:PurchaseableItem
      camera:Camera
      camera:Digital
   camera:Large-Format
   camera:Lens
   camera:Body
camera:Viewer
   camera:HQ-Viewer
```

Fig. 6. Taxonomic representation of the ontology from Fig. 5

on the data obtained from SIFO. In the next subsections we are going to show three use cases: how to use the algorithm to compute the leaves in a taxonomy, how to use it to obtain the structural similarity of two ontologies and finally how to use to align similar ontologies.

4.2.1. Using SIFO to compute the number of leaves in a taxonomy

As we commented earlier, there are techniques that compute the leaves in a graph for perfoming a graph analysis. In this sense, SIFO is easy to extend in order to compute the number of leaves in a taxonomy. To do so, it is only necessary to compute the number of the deepest taxons. Figure 8 shows how this can be done.

We have computed the leaves of the taxonomy for an example but, it is possible to compute other features such as paths. If readers think that the attributes we provide are not enough, it is easy to get others.

4.2.2. Using SIFO for comparing structural similarities

Definition 8. *We define the structural index of an ontology as a number that tells global information about the total number of children, brothers and so on.*

It is possible to use SIFO for extracting structural indexes of ontologies in order to compare its structural similarity. As we show in the state-of-the-art, some techniques use statistical methods for obtaining the structural similarity. It can be useful for adjusting the quality of the generated mappings, for example.

We have used the data structure filled by the algorithm for computing structural indexes of ontologies in several domains: bibliography, departments, genealogy and people. Then we have compared them. Table 3 shows the results we have obtained. Obviously, the higher percentage, the higher the structural similarity of the compared ontologies.

```
var max, leaves: integer
max := leaves := 0
for i := 0 to DS.size
   if (DS[i].depth > max)
      max := DS[i].depth
   end if
end for
for j := 0 to DS.size
   if (DS[j].depth = max)
      leaves++
   end if
end for
return leaves
```

Fig. 7. Computing the leaves

```
var acum : integer
acum := 0
for i := 0 to DS.size
   acum := acum + DS[i].depth
   acum := acum + DS[i].children
   acum := acum + DS[i].brothers
   acum := acum + DS[i].leftbrothers
   acum := acum + DS[i].samelevel
end for
return acum
```

Fig. 8. Portion of code to extract a structural index of the ontology

Ontologies	*Structural Similarity*	*Percentual*
Bibliography (14)*vs*(15)	515/6890	7.4%
Departments (16)*vs*(17)	4515/57380	7.8%
Genealogy (18)*vs*(19)	180/275	65.4%
People (23)*vs*(24)	525/3150	16.6%

Table 3
Comparison of indexes of structural similarity

4.3. SIFO in real alignment situations.

To obtain more accurate ontology alignments, we can combine SIFO with other algorithms. For example, the similarity between two concepts could be given by the formula:

$$similarity(c1, c2) = Levenshtein \oplus y \cdot info\ from\ SIFO$$

$$where\ y \in [0, 1]$$

This formula allow us to align ontologies from the point of view of the elements and from the ontology structure. The Levenshtein algorithm (22) calculates the string similarity, SIFO gives information about the structure and y is a corrector that depends on the trust in the similarity of domains.

We are going to use the information from SIFO for aligning (23) and (24) in the following way:

People1	People2	n
$Male$	$Divorce$	.600
$IndividualEvent$	$Marriage$	.600
$Event$	$FamilyEvent$	.600
$FamilyEvent$	$Death$	.600
$Individual$	$Birth$	.600
$Female$	$IndividualEvent$	.600
$DeathEvent$	$Family$	.600
$MarriageEvent$	$Individual$	.600

Table 4

Structural results

People1	People2	Lev	SIFO	Total
$Divorce$	$DivorceEvent$	.582	0	.582
$Marriage$	$MarriageEvent$	.615	0	.615
$FamilyEvent$	$FamilyEvent$	1.00	0	1.00
$Death$	$DeathEvent$	.500	0	.500
$Birth$	$BirthEvent$	.500	0	.500
$IndividualEvent$	$IndividualEvent$	1.00	0	1.00
$Event$	$Event$	1.00	0	1.00
$Family$	$Family$	1.00	0	1.00
$Male$	$Divorce$	.141	.600	.741
$IndividualEvent$	$Marriage$	.200	.600	.800
$FamilyEvent$	$Death$	≈ 0	.600	.600
$Individual$	$Birth$	.100	.600	.700
$Female$	$IndividualEvent$	.020	.600	.620
$DeathEvent$	$Family$	.010	.600	.610
$MarriageEvent$	$Individual$	.007	.600	.607

Table 5

Final results for the alignment.

$$If(DS.taxon.attribute = DS2.taxon.attribute) \rightarrow sim = sim + 0.2$$

Table 4 shows the results from SIFO and. Table 5 shows the total results we have obtained (Because the ontologies belongs to exactly the same domain we have chosen $y = 1$):

Finally, we have set a benchmark in order to prove the efficiency of an implementation of our algorithm. We have used a portion of the benchmark provided by the Ontology Evaluation Initiative (25) to test matching tools and have measured the times using a Intel Centrino, 1.66 GHz and 512 MB for RAM. The times have not always been the same (due to the time slice effect in monoprocessor systems), so we have listed (Table 6) ten times and we have computed the average.

4.4. Discussion

We have extended Table 7 from (26) for comparing the computation complexity of some of the best alignment tools. SIFO is not a tool, but it can be implemented in that form. Included tools are NOM (27), PROMPT (28), Anchor-PROMPT (29), GLUE (30) and QUOM (31). All of the complexity values in Table 7 are given working in the assumption that gaining access to the ontology implies a constant penalty.

As it can be appreciated, the complexity of SIFO is as good as the considered tools. Therefore, it seems to be a good idea to include it in matching tools to suplement other methods.

Our proposal, like most of solutions in the field of engineering, has several advantages, but also several disadvantages. These are some of them:

Strengths

$Ontology$	$AverageTime\ (seconds)$
101	.062
102	.312
103	.063
104	.062
201	.047
202	.062
203	.046
204	.063
224	.092
225	.032
226	.031
Bib/MIT	.093
$BibTeX/UMBC$	.031
$Karlsruhe$	.125
$INRIA$	.112

Table 6
Process time for some ontologies

Tool	**Complexity**
NOM	$O(n^2 \cdot log^2 n)$
$PROMPT$	$O(n \cdot log\ n)$
$Anchor-PROMPT$	$O(n^2 \cdot log^2 n)$
$GLUE$	$O(n^2)$
QOM	$O(n \cdot log\ n)$
SIFO	$O(n \cdot log\ n)$

Table 7
Comparing complexities

– It allows information to be obtained that can be helpful in order to take decisions in alignment scenarios. As we have shown in the example, SIFO is able to discover reasonable correspondences between ontologies to align.
– The algorithm that we propose is valid for ontology alignment, but also for aligning taxonomies or directory listings.
– Its computational complexity is the same as the most efficient alignment techniques we have studied.

Weakness

– Related to ontology alignments, it is neccesary to combine SIFO with other techniques to get satisfactory results.

Beyond the numeric results we have obtained, we have shown that to obtain new ontology matchers from the data structure that we provide is relatively easy.

As future work, we propose to improve the algorithm so that it takes into account the order of the elements. Note that some attributes like the number of brothers or children will be always identical but, for instance, the brothers on the left is an attribute which depends on the order. Two ontologies can be the same altough the taxons are not in the same order, so it is necessary to apply an extra step for ordering (in alphabetical order) the elements of the ontology. In this way, reorders of the taxons will not be able to defraud the algorithm.

5. Linguistic Matching

Nowadays there are a lot of techniques and tools for addressing the ontology alignment problem, however, the nature of the problem means existing solutions for real situations are not fully satisfactory. As a

result, the Google Similarity Distance has appeared recently. Its purpose is to mine knowledge from the Web using Google in order to obtain satisfactory alignments. This section consists of an experiment for testing not only Google, but other search engines using this similarity distance.

In fact, we are interested in three characteristics of the World Wide Web (WWW):

1. It is one of the biggest databases in the world. And possibly the most valuable source of general knowledge.
2. It is close to human language, and therefore can help to address problems related to the natural language processing.
3. It provides mechanisms to separate relevant from non-relevant information.

In this way, we believe that the most outstanding contribution of this work is the identification of the best sources of web knowledge for solving the problem of aligning ontologies precisely. In fact, in (38), the authors state: "We present a new theory of similarity between word and phrases based on information distance and Kolmogorov complexity. To fix thoughts, we used the World Wide Web (WWW) as the database, and Google as the search engine. *The method is also applicable to other search engines* and databases". Our work is about those search engines.

Under no circumstances can this work be considered as a demostration that one particular search engine is better than another or that the information it provides is more accurate.

Definition 9 (Similarity measure). *A similarity measure sm is a function* $sm : \mu_1 \times \mu_2 \mapsto R$ *that associates the similarity of two input solution mappings* μ_1 *and* μ_2 *to a similarity score* $sc \in \Re$ *in the range [0, 1].*

A similarity score of 0 stands for complete inequality and 1 for equality of the input solution mappings μ_1 and μ_2.

Definition 10 (Hit). *Hit is an item found by a search engine to match specified search conditions. More formally, we can define a hit as the function* $hit : \vartheta \mapsto N$ *which associates a natural number to a word (or set of words) to ascertain its popularity in the Internet.*

A vaule of 0 stands for no popularity and the bigger the value, bigger its associated popularity.

5.1. Design of our Experiment

To carry out our experiment, we are going to use the following method to measure the popularity of two terms: Let $c1$ and $c2$ concepts belong to two differents ontologies, let : be a string concatenation operator, then:

$$similarity(c1, c2) = \frac{\frac{hit(c1:c2)+hit(c2:c1)}{hit(c1)\times hit(c2)}}{\chi}$$

This similarity measure gives us an idea of the number of times that two concepts appear together in comparison with the number of times that the two concepts appear separately.

We have chosen arbitraruly the following search engines for testing the metric: Google[3], Yahoo[4], Lycos[5], Altavista[6], MSN[7] and Ask [8].

Some of these companies do not allow many queries to be launched in order to extract knowledge, so we have designed a small experiment: we are going to align two small ontologies from Russia (32) and (33) because some ontology matching tools have used these ontologies in the past.

[3] http://www.google.com
[4] http://www.yahoo.com
[5] http://www.lycos.com
[6] http://www.altavista.com
[7] http://www.msn.com
[8] http://www.ask.com

Russia1	Russia2	Google	Yahoo	Lycos	AltaVista	MSN	Ask
food	food	1.00	0.00	0.01	1.00	0.01	0.02
drink	drink	1.00	0.01	0.30	1.00	0.06	0.04
traveller	normal_traveller	0.00	0.00	0.00	0.00	0.00	0.00
health_risk	disease_type	0.00	0.00	0.00	0.00	0.00	0.00
time_unit	time_unit	0.00	0.00	0.00	1.00	0.00	0.00
document	document	1.00	0.00	0.01	1.00	0.01	0.02
approval	certificate	0.84	0.20	0.00	1.00	0.00	0.00
payment	means_of_payment	0.00	0.45	0.00	1.00	0.00	0.00
monetary_unit	currency	0.00	0.42	0.00	1.00	0.00	0.00
unit	unit	1.00	0.00	0.01	1.00	0.03	0.03
adventure	sport	1.00	0.01	0.03	1.00	0.04	0.40
building	public_building	0.40	0.11	0.00	1.00	0.00	0.00
flight	air_travel	0.80	0.15	0.03	1.00	0.02	0.00
river_transfer	cruise	0.00	0.12	0.00	1.00	0.00	0.00
railway	train_travel	0.00	0.98	0.00	1.00	0.00	0.00
political_area	political_region	0.00	0.40	0.00	1.00	0.00	0.00

Table 8

Results obtained from the different search engines

Russia1	Russia2	Min.	Max.	Arit.	FOAM	RiMOM
food	food	0.00	1.00	0.34	1.00	0.50
drink	drink	0.01	1.00	0.40	1.00	0.71
traveller	normal_traveller	0.00	0.00	0.00	0.00	0.00
health_risk	disease_type	0.00	0.00	0.00	0.00	0.17
time_unit	time_unit	0.00	1.00	0.17	1.00	1.00
document	document	0.00	1.00	0.34	1.00	0.99
approval	certificate	0.00	1.00	0.34	0.00	0.21
payment	means_of_payment	0.00	1.00	0.24	0.00	0.37
monetary_unit	currency	0.00	1.00	0.24	0.00	0.00
unit	unit	0.00	1.00	0.35	1.00	1.00
adventure	sport	0.01	1.00	0.41	0.00	0.01
building	public_building	0.00	1.00	0.25	0.80	0.60
flight	air_travel	0.00	1.00	0.17	0.00	0.62
river_transfer	cruise	0.00	1.00	0.19	0.00	0.21
railway	train_travel	0.00	1.00	0.33	0.00	0.54
political_area	political_region	0.00	1.00	0.23	0.00	0.40

Table 9

Comparison of the obtained mappings

5.2. Empirical Evaluation

Table 8 shows the result we have obtained.

Table 9 shows a comparison of the mappings obtained, where Min.=Minimun Max.=Maximum Arit.=Arithmetic mean. FOAM (16) and RiMOM(34) are matching tools that have provided good results for the Ontology Alignment Contest (25) in the past.

Moreover, it is important to point out that this experiment was performed in February 2008. Because information indexed by search engines is not static.

5.3. Related Work

Several authors have used Web Knowledge in their works, or rather they have used a generalization of it: background knowledge (35)(36)(37). Background knowledge involves all kind of sources for extracting

information: dictionary, thesauri, search engines, and so on, so Web Knowledge can be considered a subtype of it.

On the other hand, these works present methodologies and validations for these methodologies, but they do not provide comparison statistics. Our work can be seen as an extension of (38)(39)(40), where several formulas and mechanisms to benefit from Google knowledge are provided. Our work is similar to these studies, but whereas they focus in theoretical aspects, we have focussed on the more practical side, and we provide a statistical analysis of a larger set of web sources.

6. Statistical Matching

This section is about an experiment in which we have compared the textual rendering of ontologies in order to get more accurate alignments between them. The experiments we have performed consist on three main steps: rendering in a textual way two ontologies, comparing the obtained text with several algorithms for text comparing and, using the obtained result as a factor to improve the alignments between them. As result, we got some evidences that this technique gives us a good measure of the similarity of ontologies and, therefore can allow us to improve the effectiveness of the alignment process.

6.1. Problem Statement

Definition 11. *Textual rendering of an ontology is the result of printing the information contained in that ontology.*
It can be expressed more formally, let e an entity from an ontology O, and let $t(e)$ a function that prints the identifier of an entity, then a textual rendering T from an ontology O is an expression such:

$$\forall e \in O, \exists t(e) \Rightarrow T(O) = \{t(e)\}$$

Example 1. Textual rendering for Figure 9 is *A man is a person. A woman is a person*.

Now, we are going to explain why we think that textual renderings of ontologies are interesting.

Example 2. Note Figure 9 and Figure 10; they are very simple ontologies. They are very similar, too. For example, it is easy to align the concepts man and $woman$, using any algorithm for string matching. But, what is about $person$ and $human\ being$? We know that both represent the same object of the real world, but what computer algorithm can tell us that are the same? Based on string similarity techniques cannot. Based on taxonomy algorithms can increase the probability, but it is not enough. Based on WordNet algorithms can, but they are dangerous; imagine such concepts as 'plane' and 'aeroplane', they are synonyms, but only in some situations. We think that we can solve this problem and we are going to make

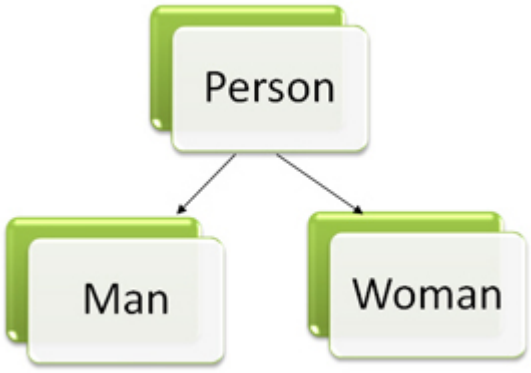

Fig. 9. Ontology sample number 1

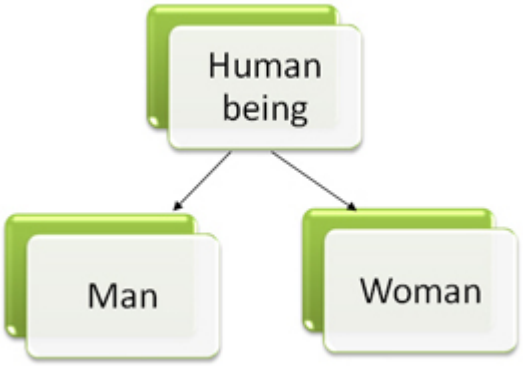

Fig. 10. Ontology sample number 2

an experiment to show it: Let's remember the textual rendering from the first ontology: *A man is a person. A woman is a person*.

On the other hand, textual rendering for the second ontology sample is: *A man is a human being. A woman is a human being*. Now, if we compare the two textual renderings using an algorithm as Loss of Information (LOI) (13), we have a 76.9 percent of similarity between them. We propose to use this result as a factor to increase the probability of the mappings in the output alignment.

In this sense, we think that we can use this observation in order to formulate a generic technique for improving ontology mappings.

The experiment that we are going to perform consists of a previous task and then three steps. The previous task is to launch a task to align the ontologies. It is interesting to launch a simple algorithm in order (as a based on similarity string algorithm) to see how much the next steps increase the quality of the alignment. Then:

1. Rendering the ontologies.
2. Comparing the obtained text.
3. Using the result as a factor to increase the probability of the mappings may be true.

Although we have defined textual rendering already, there are several ways to render the ontology in a textual way:

Definition 12. *Crude rendering is the kind of rendering that only prints the information of the concepts and properties, excluding the relations. So it loses information about the structure.* It is good when we wish to compare only the content of the ontologies.

- **Definition 12.1**. *Partial Crude rendering is a kind of rendering used to compute the similarity rate between a concrete kind of entities in two ontologies.* It is useful in cases where concepts are very similar but other entities (properties, relations, instances, so on) are very different.
- **Definition 12.2**. *Full Crude rendering is a kind rendering used to compare the contents of the whole ontologies.* It seems to be useful when compared ontologies are very closed.

Definition 13. *Full rendering is the kind of rendering which allows to rebuild the ontology because it prints information about the content and the structure. So it is a rendering without loss of information.* It is useful in order to compare not only the contents, but the structures.

- **Definition 13.1**. *Partial Full rendering prints all the information related to a kind of entities.* As we commented earlier, it is useful when concepts are closed, but we think that there are very different instances, for example.
- **Definition 13.2**. *Complete rendering prints all the information of the ontology, so the process is reversible.*

Crude renderings try to get a measure of the resemblance of the vocabularies. In full renderings, the resemblance of vocabularies is important, but each time that a entity appear we print a more elaborated message about it. Note that the message we print is similar for the two ontologies, so we are increasing the similarity between the generated text, but also reducing the importance of the vocabularies.

In order to get empirical results from our theory, we are going to perform an experiment over two public ontologies. We have chosen the ontology about bibliography of the Institute of Information Sciences (ISI) from California, USA (41). And the ontology about bibliography from the University of Yale (42), in the United States too. Originally, both ontologies were in DAML format, but we have converted them into OWL format in order to allow our software to process them. We have chosen them because we guess they have a high degree of commonality and, therefore the experiment could show us the merits of our proposal. Other important details we have considered are:

- The argument R of the mappings (relation between the entities) will be Equivalence only.
- We have determined that the degree of similarity between the textual renderings will be used for increase the n of the mappings (probability of relation between them be true).

ISI	Yale	n
$patent$	$Literal$	0.285
$collection$	$Incollection$	0.833
$collection$	$Publication$	0.545
$booklet$	$Incollection$	0.333
$booklet$	$Book$	0.428
$techreport$	$Techreport$	0.900
$phdthesis$	$Inproceedings$	0.307
$book$	$Book$	0.750
$manual$	$Literal$	0.285
$incollection$	$Incollection$	0.916
$incollection$	$Publication$	0.416
$conference$	$Incollection$	0.250
$proceedings$	$Inproceedings$	0.846
$inproceedings$	$Inproceedings$	0.923
$article$	$Article$	0.857
$inbook$	$Incollection$	0.250
$inbook$	$Book$	0.500

Table 10
Concept alignment. Threshold: 0.25

6.2. Results

1. At first time, we have performed a syntactic alignment of the ontologies. We have used the Levenshtein algorithm (22). Table 10 shows the results for the concept alignment. We have determined a low threshold for getting a significative number of pairs. Table 11 shows the results for the properties alignment. Many of them are the same in both ontologies.
2. At second time, we have performed the rendering over ontologies from the ISI and Yale. We have used Full Crude Rendering. In this way, we give more importance to the similarity of the vocabularies than to the structure of the ontologies.
3. We have used the Loss Of Information (LOI) algorithm for comparing both generated texts, we have obtained a similarity degree of 42.2 percent.
4. Finally, we have used that 42.2 percent for increase the argument n of the mappings be true (we have used the formula $n = n + (0.422 \cdot n)$. In this way, the higher values are increased significatively, while lower probabilities not. Table 12 and Table 13 shows us the new results for the concepts and the properties respectively.

ISI	Yale	n
title	*title*	1.000
title	*booktitle*	0.555
note	*note*	1.000
institution	*institution*	1.000
howpublished	*publisher*	0.667
editor	*editor*	1.000
number	*number*	1.000
author	*author*	1.000
volume	*volume*	1.000
location	*Publication*	0.636
year	*year*	1.000
publisher	*publisher*	1.000
mrnumber	*number*	0.750
annote	*note*	0.666
booktitle	*title*	0.555
booktitle	*booktitle*	1.000
edition	*editor*	0.714
organization	*Publication*	0.500
pages	*pages*	1.000
affiliation	*Publication*	0.545

Table 11
Property alignment. Threshold: 0.5

In Table 14, we have extracted a statical summary from the results of our proposal[9]

As you can see, at least in this case, we have improved the precision, we have kept the recall and, of course, we have increased the F-Measure. But there are bad news too, the number of false positives has increased. We have considered that a relation is true when its n argument is equal or greater than 0.9.

Finally, we have repeated the experiment using ontologies from other fields: academic departments, people and genealogy. As you can see in Table 15, we cannot determine any kind of relation between the improved precision and the similarity of the textual renderings, but according to the performed experiments, the technique that we propose is able to improve the precision of the mappings.

6.3. Discussion

Note that there are a lot of concepts and properties that could be aligned using a string normalization algorithm. However, there are a few couples which couldn't. For instance: $proceedings$ and $Inproceedings$, $mrnumber$ and $number$, $collection$ and $Incollection$ and so on. Therefore, the advantages are that we have into account the similarity of the ontologies for improving the mappings. In this way, we can enrich the results generated by simple methods. We provide several ways to proceed: giving more importance to the vocabulary or giving more importance to the whole ontology. Moreover, to have into account only concrete parts of the ontologies is possible. The result of our experiment tell us that it is possible to improve the precision and F-measure of the alignment process. There are some disadvantages too; it is necessary to combine this technique with other ones, that it is to say, it is not good enough as to

[9]We have used the following formulas for the calculations:

$$Precision = \frac{Correct\ relations}{Correct\ relations + Incorrect\ relations}$$

$$Recall = \frac{Correct\ relations}{Correct\ relations + Not\ found\ relations}$$

$$F - Measure = \frac{2 \cdot precision \cdot recall}{precision + recall}$$

ISI	Yale	n (Improved)
patent	*Literal*	0.405
collection	*Incollection*	**1.000**
collection	*Publication*	0.774
booklet	*Incollection*	0.473
booklet	*Book*	0.608
techreport	*Techreport*	**1.000**
phdthesis	*Inproceedings*	0.436
book	*Book*	**1.000**
manual	*Literal*	0.405
incollection	*Incollection*	**1.000**
incollection	*Publication*	0.591
conference	*Incollection*	0.355
proceedings	*Inproceedings*	**1.000**
inproceedings	*Inproceedings*	**1.000**
article	*Article*	**1.000**
inbook	*Incollection*	0.355
inbook	*Book*	0.711

Table 12

Improved Concept alignment. Threshold: 0.25

ISI	Yale	n (Improved)
title	*title*	1.000
title	*booktitle*	0.788
note	*note*	1.000
institution	*institution*	1.000
howpublished	*publisher*	**0.946**
editor	*editor*	1.000
number	*number*	1.000
author	*author*	1.000
volume	*volume*	1.000
location	*Publication*	0.903
year	*year*	1.000
publisher	*publisher*	1.000
mrnumber	*number*	**1.000**
annote	*note*	**0.946**
booktitle	*title*	0.788
booktitle	*booktitle*	1.000
edition	*editor*	1.000
organization	*Publication*	0.710
pages	*pages*	1.000
affiliation	*Publication*	0.774

Table 13

Improved Property alignment. Threshold: 0.5

	Before	Later
Precision	63.1%	79.1%
Recall	92.3%	92.3%
F − Measure	74.9%	86.5%

Table 14

Summary from the experiment

Ontologies	Similarity	Precision
$Departments$ [40] vs [41]	14.8%	+12.5 $p.p.$
$People$ [23] vs [24]	19.2%	+8.3 $p.p.$
$Bibliography$ [42] vs [43]	42.2%	+16.0 $p.p.$
$Genealogy$ [44] vs [45]	61.2%	+7.6 $p.p.$

Table 15
Results obtained from alignments in other domains

generate good mappings by itself. Besides, it increases the number of false positives. On other hand, you may wondered why we have not improved the recall. Think that we improve existing results, we do not look for new ones. We increase the probabilities of the relations be true, as higher are these probabilities, more be incremented and vice versa. But, we do not launch a alignment task again. In the experiments, we have obtained a good degree of similarity, we think that this result means that compared ontologies are similar, but we knew that we have been aligned closed ontologies. We have to study this detail more in depth in order to formulate a more accurate methodology. In this work, we have proposed a technique for getting more accurate ontology alignments. This technique is based on the comparison of the textual renderings of the ontologies to align. According to the experiments we have performed, we can conclude that comparing the textual rendering of the ontologies to align is able to improve the precision of the alignment process. However, there is work to do: At first time it is necessary to test a bigger quantity of ontologies, we are going to test the benchmark provided by the Ontology Alignment Evaluation Initiative (OAEI) (25). Moreover, it is important to determine clearly what kind of rendering is more appropriate according to the situation, and what are the best algorithms for comparing the text obtained from the textual rendering. In this way, we wish to use not only LOI algorithm, but other text metrics.

7. Meta-matching Techniques

What exactly is ontology Meta-Matching in practice? *It is the technique of selecting the appropiate algorithms, weights and thresholds in ontology alignment scenarios*. Figure 11 shows a diagram for modelling the actions in a Meta-Matching process.

Note that algorithms do not need to be reconsidered. The idea is to provide all possible algorithms initially and then automatically associate a weight of 0 to those that are not useful for solving the problem. How the algorithms are used or the weights and the threshold recalculated are what makes one Meta-Matching strategy better than another, in terms of accuaracy and time consumption.

In general, we can describe the followings as characteristics from a Meta-Matching task:

- It is not necessary for it to be done at runtime.
- It must be an automatic process.
- It must return the best possible matching function.

Moreover, Meta-matching can be seen from two points of view: (i) From the point of view of the algorithmic techniques used to obtain the matching function:

- **Aggregation.** This technique (47) determines the upper bound T(n) on the values obtained from a sequence of n matchers, then calculates the average value to be T(n)/n.
- **Combination.** The main idea is to combine similarity values predicted by multiple matchers to determine correspondences between ontology entites. Where combinations can be as simple as: arithmetic or harmonic means, maximum, minimun, Minkowski distances, any kind of weighted product or sum, and so on, or more complex combinations like Linguistic Combinations (48)
- **Composition.** Let $f_1, f_2, ..., f_n$ be n unique matchers, a composition is a function $f(O_1, O_2) = f_1 \circ f_2 \circ ... \circ f_n$. Thus, the idea of this mechanism is to use simple functions to build more complicated ones

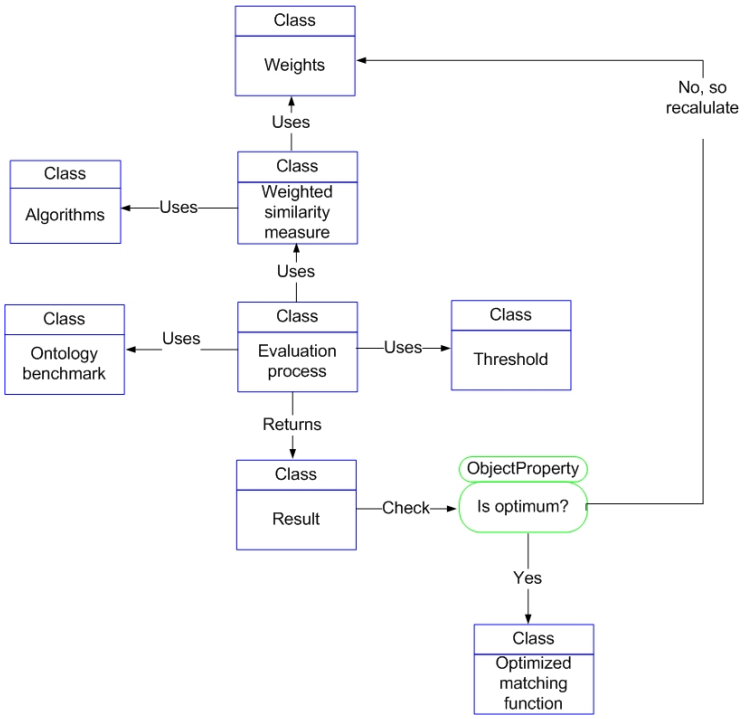

Fig. 11. General model for Meta-Matching

(ii) From the point of view of the computer science paradigm that makes the Meta-Matching possible, i.e. the form of recalculating the parameters. Although, this problem can be solved trivially by a brute force search when the number of matchers to use is low, Meta-Matching scales better for a higher number of matchers. For this reason we do not include brute force methods as a viable technique. These are the two main groups of techniques considered:

- **Heuristic Meta-Matching**, where the most outstanding approaches are Based on Genetic Algorithms meta-matching. In such case, it is said that parameters are optimized and, Greedy meta-matching, in such case, it is said that parameters are estimated.
- **Based on Machine Learning meta-matching**, where the most outstading approaches are Relevance Feedback and Neural networks training for meta-matching. In both cases, it is said that parameters are learned.

7.1. Heuristic Meta-Matching

A heuristic is a method to help to solve a problem, commonly informal. It is particularly used for a method that may lead to a solution which is usually reasonably close to the best possible answer.

Two fundamental goals in computer science are to find algorithms with *probably* good run times and with *probably* good or optimal solution quality. A heuristic is an algorithm that abandons one or both of these goals; for example, it usually finds pretty good solutions, but there is no proof that the solutions could not get arbitrarily bad; or it usually runs reasonably quickly, but there is no argument that this will always be the case.

Therefore, the use of heuristics is very common in real world implementations. For many practical problems, a heuristic algorithm may be the only way to get good solutions in a reasonable amount of time.

A lot of tools clearly implements heuristic Meta-Matching, we can see the most clear example in the default configuration of COMA (13), where an expert has adjusted initially the weights of the conceptual and structural techniques respectively. In order to avoid the human interaction in this field, we can use Genetic Algorithms for optimizing the parametrers or Greedy Algorithms to estimate them.

7.1.1. Based on Genetic Algorithms methods

Genetic Algorithms (GAs) (49) are adaptive heuristic search algorithm premised on the evolutionary ideas of natural selection and genetic. The basic concept of GAs is designed to simulate the natural evolutionary system.

This approach that is able to work with several goals (50): maximizing the precision, maximizing the recall, maximizing the f-measure or reducing the number of false positives. Moreover, it has been tested combining some leading-edge similarity measures over a standard benchmark and the results obtained show several advantages.

Another proposal is (51), a genetic algorithm-based optimization procedure for ontology matching problem that is presented as a feature-matching process. First, from a global view, we model the problem of ontology matching as an optimization problem of a mapping between two compared ontologies, and every ontology has its associated feature sets. Secondly, as a powerful heuristic search strategy, a genetic algorithm is employed for the ontology matching problem. Given a certain mapping as optimizing object for GA, a fitness function is defined as a global similarity measure function between two ontologies based on feature sets.

7.1.2. Greedy Meta-Matching

Greedy (52) Meta-Matching is a technique which, given a particular matching task, tries to automatically tune an ontology matching function. For that purpose, it tries to choose the best matchers and parameters to be used, but with a short-sighted strategy. The most popular example of this technique can be found at (53). Results from Greedy techniques are, in general, worse than those based on Genetics, but its computation time also tends to be much lower.

7.2. Based on Machine Learning methods

Based on Machine Learning (54) Meta-Matching techniques considers both schema information and instance data. This kind of Meta-Matching can be divided into two subtypes[10]:

7.2.1. Relevance Feedback.

This kind of approaches explores the user validation of initial ontology alignments for automatically optimising the configuration parameters of the matching strategies. A clear example of this kind of Meta-Matching is (55). Using such techniques we are able to avoid the user, and maybe the context, the depedency of the matching task, however, it implies spending much time on training the systems. To do that automatically, it is possible to use Neural Networks.

7.2.2. Neural Networks Training for Meta-Matching.

A neural network (56) is an interconnected group of artificial neurons that uses a mathematical or computational model for information processing based on a connectionistic approach to computation. In most cases a neural network is an adaptive system that changes its structure based on external or internal information flowing through the network. In more practical terms neural, networks are non-linear statistical data modeling or decision making tools. They can be used to model complex relationships between inputs and outputs or to find patterns in data.

Neural networks training for Meta-Matching consist of training a neural network with heterogeneous enough benchmarks and then using the knowledge to predict new similarity functions. This is the case of (57) where authors exploit an approach to learn the weights for different semantic aspects of ontologies, through applying an artificial neural networks technique.

Another approach consists of an automatic ontology alignment method based on the recursive neural network model that uses ontology instances to learn similarities between ontology concepts. Recursive neural networks are an extension of common neural networks, designed to process efficiently structured data (58).

[10] Although learning techniques exist such as Bayes learning, WHIRL learning, decision trees or stacked generalisation, there are no Meta-Matching proposals using them as yet

8. Greedy Strategy

In this section, we are going to discuss the greedy strategy to solve the Meta-Matching problem. Moreover, we propose an efficient greedy algorithm and compute its associated complexity according to the O notation.

8.1. Maximum Similarity Measure

An inital approach for an ideal Customizable Similarity Measure which would be defined in the following way:

Let $\vec{A}$ be a vector of atomic matching algorithms in the form of a similarity measure and $\vec{w}$ a numeric weight vector then:

$$MaSiMe(c1, c2) = x \in [0, 1] \in \Re \rightarrow \exists \left\langle \vec{A}, \vec{w} \right\rangle, x = max(\sum_{i=1}^{i=n} A_i \cdot w_i)$$
$$\textit{with the following restriction } \sum_{i=1}^{i=n} w_i \leq 1$$

But from the point of view of engineering, this measure leads to an optimization problem for calculating the weight vector, because the number of candidates from the solution space is infinte. For this reason, we present MaSiMe, which uses the notion of granularity for setting a finite number of candidates in that solution space. This solution means that the problem of computing the similarity can be solved in a polinomial time.

Definition 14. Maximum Similarity Measure (MaSiMe).

$$MaSiMe(c1, c2) = x \in [0, 1] \in \Re \rightarrow \exists \left\langle \vec{A}, \vec{w}, g \right\rangle, x = max(\sum_{i=1}^{i=n} A_i \cdot w_i)$$
$$\textit{with the following restrictions } \sum_{i=1}^{i=n} w_i \leq 1 \wedge \forall w_i \in \vec{w}, w_i \in \{\dot{g}\}$$

Example 3. Given an arbitrary set of algorithms and a granularity of 0.05, calculate MaSiMe for the pair *(author, name_author)*.

$$MaSiMe(author, name\_author) = .542 \in [0, 1] \rightarrow$$
$$\exists \langle A = (L, B, M, Q), w = (0.8, 0, 0, 0.2), g = 0.05 \rangle, 0.542 = max(\sum_{i=1}^{i=4} A_i \cdot w_i)$$

Where L = Levhenstein (22), B = BlockDistance (13), M = MatchingCoefficient (13) , Q = QGramsDistance (58)

There are several properties for this definition:

Property 1 (Continiuous uniform distribution). A priori, MaSiMe presents a continuous uniform distribution in the interval $[0, 1]$, that is to say, its probability density function is characterised by

$$\forall\, a, b \in [0, 1] \rightarrow f(x) = \tfrac{1}{b-a}\ for\ a \leq x \leq b$$

Property 2 (Maximality). If one of the algorithms belonging to the set of matching algorithms returns a similarity of 1, then the value of MaSiMe is 1.

$$\exists A_i \in \vec{A},\ A_i(c1, c2) = 1 \rightarrow MaSiMe(c1, c2) = 1$$

Moreover, the reciprocal is true

$$MaSiMe(c1, c2) = 1 \rightarrow \exists A_i \in \vec{A},\ A_i(c1, c2) = 1$$

Example 4. Let us suppose that we have: $\vec{A}$ = *(Google Similarity Distance (38), BlockDistance, MatchingCoefficient, QGramsDistance)* and $g = 0.05$, calculate $\vec{w}$ for maximizing R in the mapping *(plane, aeroplane, Equivalence, R)*
Solution:

$$(1, 0, 0, 0)$$

So the optimum hybrid matcher for the equivalence of *(plane, aeroplane)* is:

$1 \cdot GoogleDistance + 0 \cdot BlockDistance + 0 \cdot MatchingCoefficient + 0 \cdot QGramsDistance, R = 0.555$

Moreover, we can say that the worst vector is $\vec{w} = (0, 0.8, 0.15, 0.05)$ because it generates a $R = 0.027$

Property 3 (Monoticity). Let S be a set of matching algorithms, and let S' be a superset of S. If MaSiMe has a specific value for S, then the value for S' is either equal to or greater than this value.

$$\forall S' \supset S, MaSiMe_s = x \rightarrow MaSiMe_{s'} \geq x$$

Property 4. (Dependent completeness). If one of the algorithms belonging to the set of matching algorithms provides a similarity of 1 and the chosen granularity is not a submultiple of 1, then the value of MaSiMe is less than 1.

$$\exists A_i \in \vec{A} \wedge\ 1 \notin \{\dot{g}\} \wedge A_i(c1, c2) = 1 \rightarrow MaSiMe(c1, c2) < 1$$

Example 5. Let us suppose we have the same conditions as the Example 4, i.e., that we have: *A = (Google Similarity Distance, BlockDistance, MatchingCoefficient, QGramsDistance)* but now $g = 0.21$. Calculate $\vec{w}$ for maximizing R in the mapping *(plane, aeroplane, Equivalence, R)*
Solution:

$$(0.84, 0, 0, 0)$$

So the optimum hybrid matcher for the equivalence for *(plane, aeroplane)* is not the same as that in Example 4.

The reason is that if the granularity is not multiple of 1, the summation from the weight vector cannot be 1, and therefore $\vec{A} \cdot \vec{w}$ cannot be optimal.

8.2. Computing the Weight Vector

Once the problem is clear and the parameters $\vec{A}$ and g are known, it is neccesary to effectively compute the weight vector. At this point, we leave the field of similarity measures to move into the field of enginering.

It is possible to compute MaSiMe in several ways, for this work, we have designed a greedy mechanism that seems to be effective and efficient. The next paragraphs discuss this mechanism.

8.2.1. Greedy Strategy.

A greedy strategy follows the problem solving heuristic of making the locally optimum choice at each stage with the hope of finding the global optimum.

Let S the set of all the ontology matching algorithms

$$\exists A \subset S, \exists g \in [0, 1] \in Q, \forall i, j, k, ..., t \in \{\dot{g}\} \rightarrow \exists \vec{r}, r_i = \vec{A} \cdot (i, j - i, k - j, ..., 1 - t)$$
$$\textit{with the followings restrictions } j \geq i \wedge k \geq j \wedge 1 \geq k$$
$$R = max\ (r_i) \leq 1$$

Where,

- g is the granularity
- $(i, j-i, k-j, ..., 1-t)$ is the pattern for the weight vector
- r_i are the partial results of the measure
- R is the maximum of the partial results, and therefore the value for MaSiMe

The algorithm can be suspended when it obtains a partial result equal to 1, because this is the maximum value than we can hope for.

8.2.2. Complexity.

The strategy seems to be brute force, but it is not. Have into account that the input data size is $n^{lenght\ of\ \vec{A}}$, but the computational complexity for the algorithm according to O notation is

$$O(n^{lenght\ of\ \vec{A-1}})$$

In this way, the total complexity (TC) for MaSiMe is:

$$TC(MaSiMe_A) = O(max(max(O(A_i)), O(strategy)))$$

and therefore for MaSiMe using the greedy strategy

$$TC(MaSiMe_A) = O(max(max(O(A_i), O(n^{lenght\ of\ A-1}))))$$

Example 6. Given the set *A = {Levhenstein, BlockDistance, MatchingCoefficient, QGrams-Distance}* , the complexity for the matching process using MaSiMe is calculated.

$$TC(MaSiMe_A) = O(max(O(n^2), O(n^3))) = O(n^3)$$

8.3. Empirical evaluation

In this section, we test an implementation of MaSiMe. We have configured MaSiMe in the following way: For the matching algorithms vector, we have chosen an arbitrary set of algorithms *A = {Levhenstein (22), Stolios (59), SIFO (60), Google (38) (39), Q-Gram (58) }* and for granularity, g = 0.05. Moreover, the threshold for relevant mappings is 0.9, i.e. all mappings with a probability greater than 0.9, will be included in the output alignment.

Before testing MaSiMe over a standard benchmark, we show an example of mappings that MaSiMe has been able to discover from ontologies (17) and (18). It is an arbirtrary example, but it gives us a good idea of the Measure. We have compared it to two of the most outstanding tools for ontology alignment: FOAM (16) and RiMOM (34).

We have used the default configuration for these tools, but not for MaSiMe, where the notion of configuration does not exist. Table 16 shows the results:

9. Genetic Strategy

Genetic Algorithms (GAs) are often used to search along very high dimensional problems spaces. For example, if we want to find the maximum value of the function wsf with three independent variables x, y and z:

$$wsf(O_1, O_2) = x \cdot datatype(O_1, O_2) + y \cdot normalization(O_1, O_2) + z \cdot synonyms(O_1, O_2)$$

Russia1	Russia2	FOAM	RiMOM	MaSiMe
food	food	1.00	0.50	1.00
drink	drink	1.00	0.71	1.00
traveller	normal_traveller	0	0	0.90
health_risk	disease_type	0	0.17	0.17
time_unit	time_unit	1.00	1.00	1.00
document	document	1.00	0.99	1.00
approval	certificate	0	0.21	0.96
payment	means_of_payment	0	0.37	0.86
monetary_unit	currency	0	0	0.19
inhabitant	cititzen_of_russia	0	0.11	0.11
unit	unit	1.00	1.00	1.00
adventure	sport	0	0.01	0.10
building	public_building	0.80	0.60	0.90
flight	air_travel	0	≈ 0	0.62
river_transfer	cruise	0	0.21	0.21
railway	train_travel	0	0.54	1.00
political_area	political_region	0	0.40	0.84

Table 16
Comparison of several mappings from several tools

where x, y and z are weights to determine the importance of the three associated similarity measures, which belong, for instance, to the continous interval [0, 1]. The problem that we want to solve is finding a good value of x, y and z to find the largest possible value of wsf.

While this problem can be solved trivially by a brute force search over the range of the independent variables x, y and z, the GA method scales very well to similar problems of a higher dimensionality; for example, we might have functions using a large number of independent variables x, y, z,..., t. In this case, an exhaustive search would be prohibitively expensive.

The methodology of the application of a GA requires defining the following strategies:

- Characterize the problem by enconding in a string of values the contents of a tentative solution.
- Provide a numeric fitness function that will allow to rate the relative quailty of each individual tentative solution in a population.

Out first task is to characterize the search space as some parameters. We need to encode several parameters in a single chromosome, so we have designed a method for converting a 10-bit representation to a set of floating-point numbers in the arbitrary range [0, 1].

Later, we haved designed a fitness function to determine which chromosomes in the population are most likely to survive and reproduce using genetic crossover and mutation operations.

For the returning of the fitness function, we can choose any parameter provided for the alignment evalution process, thus, precision, recall, f-measure or fall-out. In this way, we are providing the possibility to select one of these goals.

- Optimizing the precision
- Optimizing the recall
- Optimizing the f-measure
- Reducing the number of false positives

All of them are concepts used in Information Retrieval (62) for measuring the goodness of a retrieval task. Precision is the percentage of items returned that are relevant. Recall is the fraction of the items that are relevant to a query (in this case, to a matching task). F-measure is a weighted sum from precision and recall. Finally, false positives are relationships which have been provided to the user although they are false. In some domains, (for instance in Medicine) false positives are absolutely unwanted.

9.1. Preeliminary Study

We are going to do a preeliminary study of the parameters for the algorithm.

- For the number of genes per chromosome we have selected such values as 5, 10 and 20. A study using a T-Test distribution have shown us that that the differences between samples are not statistically significant. Therefore, we have selected 20 genes per chromosome.
- For the number of individuals in the population, we have selected such values as 20, 50 and 100. Again, a T-Test stastistical distribution have shown that the differences between these samples are not statistically significant. We have selected a population of 100 individuals.
- Related to crossover and mutation fraction, we have choosen a high value for the crossover between genes and, a little percent for mutations, because we wish a classical configuration for the algorithm.
- After ten independent executions, we noticed that the genetic algorithm do not improve the results beyond the fifth generation, so we have set a limit of five generations.

9.2. Main Experiment

Related to the conditions of the experiment, we have used:

- As similarity measure vector:
  {*Levhenstein*(22), *SIFO*(60), *Stolios*(59), *QGrams*(58)}
- The GA has been configured having into account the following parameters[11]:
  - 20 genes per chromosome
  - A population of 100 individuals
  - 0.98 for crossover fraction
  - 0.05 for mutation fraction
  - We allow 5 generations
- The platform characteristics: Intel Core 2 Duo, 2.33Ghz and 4GB RAM. The developing language was Java.

10. Evaluation

The evaluation of a Meta-Matching strategy consists of the evaluation of its returned matching function. There are several ways to evaluate an output alignment:

- Compliance measures provide some insight on the quality of identified alignments.
- Performance measures show how good the approach is in terms of computational resources.
- User-related measures help to determine the overall subjective user satisfaction, partially measured, e.g., through user effort needed.
- There are task-related measures, which measure how good the alignment was for a certain use case or application.

In practice, however, there is a degree of agreement to use some measures from the Information Retrieval field (61). These are: *precision*, *recall*, *f-measure* and *fall-out*.

$$Precision = \frac{\{relevant\ mappings\} \cap \{retrieved\ mappings\}}{\{relevant\ mappings\}}$$

$$Recall = \frac{\{relevant\ mappings\} \cap \{retrieved\ mappings\}}{\{retrieved\ mappings\}}$$

$$F - Measure = \frac{2 \times precision \times recall}{precision + recall}$$

[11] Fitness and search space have been explained in the previous section

Ontology	Comment	Precision	Recall	F-Meas.	Fall-out
101	Reference alignment	1.00	1.00	1.00	0.00
102	Irrelevant ontology	N/A	N/A	N/A	N/A
103	Language generalization	1.00	1.00	1.00	0.00
104	Language restriction	1.00	1.00	1.00	0.00
201	No names	1.00	1.00	1.00	0.00
202	No names, no comments	1.00	1.00	1.00	0.00
203	Comments was missspelling	1.00	1.00	1.00	0.00
204	Naming conventions	1.00	0.91	0.95	0.00
205	Synonyms	1.00	0.19	0.33	0.00
206	Translation	1.00	0.19	0.33	0.00
221	No specialisation	1.00	1.00	1.00	0.00
222	Flatenned hierarchy	1.00	1.00	1.00	0.00
223	Expanded hierarchy	1.00	1.00	1.00	0.00
224	No instance	1.00	1.00	1.00	0.00
225	No restrictions	1.00	1.00	1.00	0.00
301	Real: BibTeX/MIT	0.93	0.23	0.37	0.06

Table 17
Behaviour of MaSiMe for the standard benchmark of the OAEI

Ontology	Comment	Precision	Recall	F-Meas.	Fallout
101	Reference alignment	1.00	1.00	1.00	0.00
102	Irrelevant ontology	N/A	N/A	N/A	N/A
103	Language generalization	1.00	1.00	1.00	0.00
104	Language restriction	1.00	1.00	1.00	0.00
201	No names	1.00	1.00	1.00	0.00
202	No names, no comments	1.00	1.00	1.00	0.00
203	Comments was missspelling	1.00	1.00	1.00	0.00
204	Naming conventions	1.00	1.00	1.00	0.00
205	Synonyms	1.00	0.71	0.83	0.06
206	Translation	1.00	1.00	1.00	0.00
221	No specialisation	1.00	1.00	1.00	0.00
222	Flatenned hierarchy	1.00	1.00	1.00	0.00
223	Expanded hierarchy	1.00	1.00	1.00	0.00
224	No instance	1.00	1.00	1.00	0.00
225	No restrictions	1.00	1.00	1.00	0.00
301	Real: BibTeX/MIT	0.90	0.69	0.78	0.07

Table 18
Behaviour of the Genetic Algorithm for the standard benchmark of the OAEI

$$Fall - out = \frac{\{non\ relevant\ mappings\} \cap \{retr.\ mappings\}}{\{non\ relevant\ mappings\}}$$

Table 17 shows the results we have obtained for the greedy strategy. Table 18 shows the results we have obtained for the genetic strategy. Figure 12 shows a graphical comparative between the two strategies we have used.

11. Related Work

If we look at literature, we can distinguish between individual ontology matching algorithms (i.e. FCA-MERGE (63) or S-Match (64)) applying only a single method of matching items i.e. linguistic or taxonomical matchers and combinations of the former ones, which intend to overcome their limitations by proposing hybrid and composite solutions. A hybrid approach follows a black box paradigm, in which

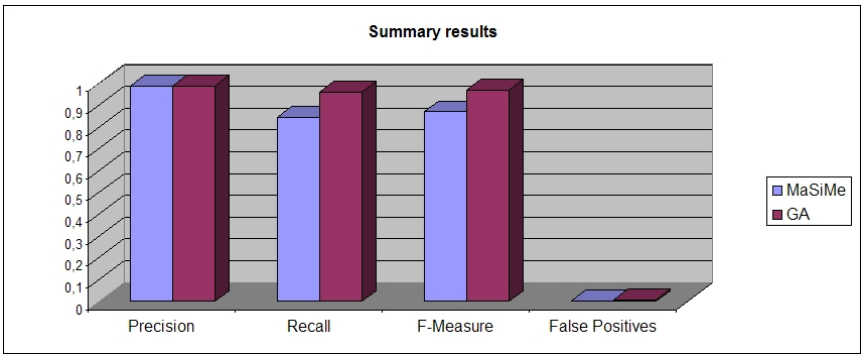

Fig. 12. Comparative results between strategies

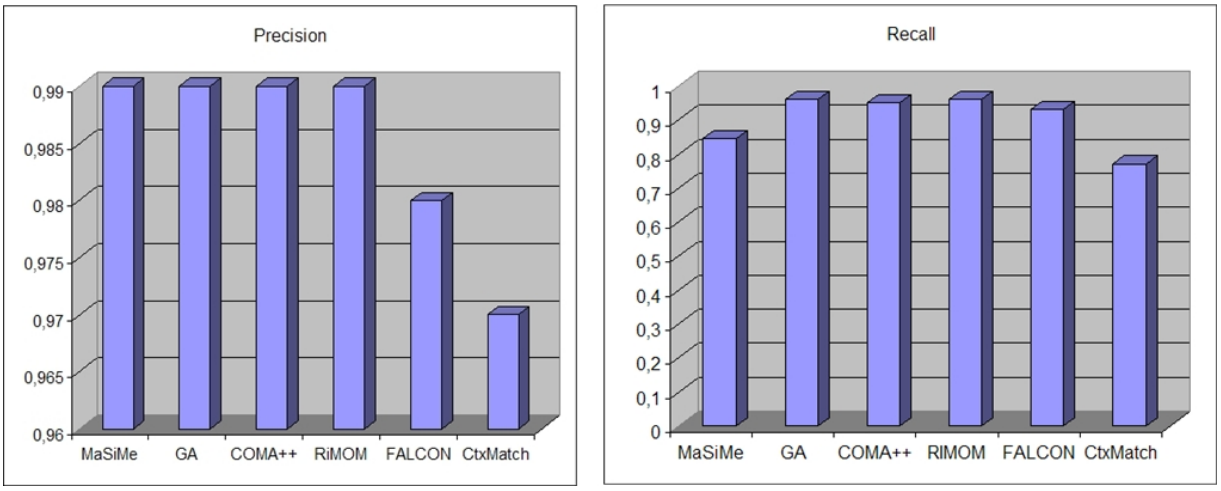

Fig. 13. Comparison with other tools

various individual matchers are melt together to a new algorithm, while the so-called composite matchers allow an increased user interaction (i.e. COMA++ (13), RIMOM (22), FALCON (65) and CtxMatch (66)).

The problem is that those kind of proposals use weights defined by an expert for configuring the matchers, but using our approaches involves to compute the weigths in an automatic way, so the process can be more accurate and faster.

To avoid the human expert intervention, there are two research lines now; one for evaluating the results of an alignment tool and maybe feedback the process (67) and another called Ontology Meta-Matching (2) that tries to optimize automatically the parameters related to matching task. So, our approach could be considered a mechanism for Meta-Matching. Most outstanding examples for this paradigm are: (i) Based on Exhaustive search solutions, (ii) Based on Neural Networks solutions, and (iii) Based on Genetic Algorithms solutions:

11.1. Based on Exhaustive Search Solutions

Ontology Meta-Matching can be solved trivially by an exhaustive search when the number of similarity measures is low, the most oustanding approach in this sense is eTuner (54) that it is a system which, given a particular matching task, automatically tunes an ontology matching system (computing one-to-one alignments). For that purpose, it chooses the most effective basic matchers, and the best parameters to be used.

However, exhaustive searches are very expensive, and unworkable when combining a great number of measures, from a computational point of view. In this sense, most of solutions try to avoid this kind of methods.

	Precision	Recall
GAOM	0.94	0.87
GOAL	**0.99**	**0.96**

Table 19
Comparison between GAOM and our proposal

11.2. Based on Machine Learning Solutions

Based on Machine Learning Meta-Matching techniques can be divided into two subtypes: Relevance feedback (68) and Neural Networks (69):

- The idea behind relevance feedback (68) is to take the results that are initially returned from a given query and to use information about whether or not those results are relevant to perform a new query: APFEL (Alignment Process Feature Estimation and Learning) is a machine learning approach that explores user validation of initial alignments for optimising automatically the configuration parameters of some of the matching strategies of the system, e.g., weights, thresholds, for the given matching task.
- Neural Networks (69) are non-linear statistical data modeling or decision making tools. They can be used to model complex relationships between inputs and outputs or to find patterns in data. SFS (70) It is a tool for ontology Meta-Matching that tries to obtain automatically a vector of weights for different semantic aspects of a matching task, such as comparison of concept names, comparison of concept properties, and comparison of concept relationships. To do that, it uses the neural networks technique.

However, these kind of solutions implies spending much effort and time on training the systems in relation to our two proposals.

11.3. Based on Genetic Algorithms Solutions

In relation to Based on Genetic Algorithm solutions, the most oustandig tool is GAOM which is a genetic algorithm based approach for solving the ontology matching problem. For the purposes of better and the more precise representation of ontology feature, it defines ontoly features from two aspects: intensional and extensional. On the other hand, ontology matching problem is modeled as a global optimization of a mapping between two ontologies. Then a genetic algorithm is used to achieve an approximate optimal solution.

Table 19 shows a comparative for the results for both GAOM and our proposal.

Although we follow the same paradigm, our proposal is slightly better in terms of numbers to GAOM as the results shows.

12. Conclusions

We have presented ontology Meta-Matching, as a novel computational discipline for flexible and accurate automatic ontology matching that generalizes and extends previous proposals for exploiting simple ontology matchers. We have presented the main techniques for ontology Meta-Matching. These techniques take into account that it is not trivial to determine what the weights of the semantic aspects should be and tries to avoid the research work depending a lot on human heuristics.

We provided an analysis of the most popular algorithms and techniques for simple matching, and characterized their relative applicability as black boxes in a Meta-Matching enviroment. It is neccessary to bear in mind that the success of the Meta-Matching depends largely on the kind of the underlying simples matchers used and the heterogeneity and soundness of the benchmarks for estimating, optimizing or learning the parameters.

We showed the most promising tools in the area of Meta-Matching. Like techniques, tools can be classified into heuristic or learning-based one. Such tools represent a serious effort to make the task of ontology matching a more independent process from users, context, and even data involved.

The lessons learned on Ontology Meta-Matching will allow us to work with other kinds of conceptual schemas for modelling knowledge (71). In this sense, we are convinced that Ontology Meta-Matching is a perfect candidate to take users a step further the state-of-the-art in terms of interoperability in the Semantic Web.

13. Acknowledgments

We thank anonymous reviewers for their very useful comments and suggestions. We thanks to Enrique Alba his useful collaboration in the part related to Genetic Algorithms. This work has been partially supported by the ICARO Project Grant, TIN2005-09098-C05-01 (Spanish Ministry of Education and Science), and Applied Systems Biology Project, P07-TIC-02978 (From Consejeria de Innovacion, Ciencia y Empresa belonging to the regiona goverment of Andalucia).

References

[1] Jorge Martinez-Gil: Thinking on the Web: Berners-Lee, Gödel and Turing. Comput. J. 50(3): 371-372 (2007).
[2] Jerome Euzenat, Pavel Shvaiko. Ontology Matching. Springer-Verlag, 2007.
[3] Philip A. Bernstein, Sergey Melnik: Meta Data Management. ICDE 2004: 875.
[4] Bin He, Kevin Chen-Chuan Chang: Making holistic schema matching robust: an ensemble approach. KDD 2005: 429-438.
[5] Marc Ehrig, York Sure: Ontology Mapping - An Integrated Approach. ESWS 2004: 76-91.
[6] Liliana Cabral, John Domingue, Enrico Motta, Terry R. Payne, Farshad Hakimpour: Approaches to Semantic Web Services: an Overview and Comparisons. ESWS 2004: 225-239.
[7] A. Prasad Sistla, Clement T. Yu, R. Venkatasubrahmanian: Similarity Based Retrieval of Videos. ICDE 1997: 181-190.
[8] Christoph Kiefer, Abraham Bernstein, Markus Stocker: The Fundamentals of iSPARQL: A Virtual Triple Approach for Similarity-Based Semantic Web Tasks. ISWC/ASWC 2007: 295-309.
[9] Jorge Martinez-Gil, Enrique Alba, Jose F. Aldana-Montes: Optimizing Ontology Alignments by Using Genetic Algorithms. NatuReS 2008.
[10] Gonzalo Navarro: A guided tour to approximate string matching. ACM Comput. Surv. 33(1): 31-88 (2001).
[11] WordNet. http://wordnet.princeton.edu. Visit date: 11-march-2008.
[12] Patrick Ziegler, Christoph Kiefer, Christoph Sturm, Klaus R. Dittrich, Abraham Bernstein: Detecting Similarities in Ontologies with the SOQA-SimPack Toolkit. EDBT 2006: 59-76.
[13] Hong Hai Do, Erhard Rahm: COMA - A System for Flexible Combination of Schema Matching Approaches. VLDB 2002: 610-621.
[14] David Aumueller, Hong Hai Do, Sabine Massmann, Erhard Rahm: Schema and ontology matching with COMA++. SIGMOD Conference 2005: 906-908.
[15] Christian Drumm, Matthias Schmitt, Hong Hai Do, Erhard Rahm: Quickmig: automatic schema matching for data migration projects. CIKM 2007: 107-116.
[16] Marc Ehrig, York Sure: FOAM - Framework for Ontology Alignment and Mapping - Results of the Ontology Alignment Evaluation Initiative. Integrating Ontologies 2005.
[17] Yannis Kalfoglou, W. Marco Schorlemmer: IF-Map: An Ontology-Mapping Method Based on Information-Flow Theory. J. Data Semantics 1: 98-127 (2003).
[18] Haggai Roitman, Avigdor Gal: OntoBuilder: Fully Automatic Extraction and Consolidation of Ontologies from Web Sources Using Sequence Semantics. EDBT Workshops 2006: 573-576.
[19] Jayant Madhavan, Philip A. Bernstein, Erhard Rahm: Generic Schema Matching with Cupid. VLDB 2001: 49-58.
[20] Kewei Tu, Yong Yu: CMC: Combining Multiple Schema-Matching Strategies Based on Credibility Prediction. DASFAA 2005: 888-893.
[21] Alexander Maedche, Boris Motik, Nuno Silva, Raphael Volz: MAFRA - A MApping FRAmework for Distributed Ontologies. EKAW 2002: 235-250.
[22] V. Levenshtein. Binary Codes Capable of Correcting Deletions, Insertions and Reversals. *Soviet Physics-Doklady*, Vol. 10, pages 707-710, August 1966.
[23] Ontology for describing an individual. http://daml.umbc.edu/ontologies/ittalks/person. Visit date: 28-jul-2008.
[24] Ontology that describe data from a person. http://www.cs.umd.edu/projects/plus/DAML/onts/ personal1.0.daml. Visit date: 28-jul-2008.
[25] Ontology Evaluation Initiative. http://oaei.ontologymatching.org. Visit date: 30-jul-2008.
[26] Marc Ehrig: Ontology Alignment: Bridging the Semantic Gap. (Contents) 2007, Springer, ISBN 978-0-387-36501-5.

[27] Marc Ehrig and York Sure. Ontology mapping - an integrated approach. In ChristophBussler, John Davis, Dieter Fensel, and Rudi Studer(Eds.): *Proceedings of the 1st ESWS*, LCNS 3053, pages 76Ű91, Heraklion (GR), Springer-Verlag, 2004.
[28] Natalya Noy and Mark Musen. PROMPT: Algorithm and Tool for Automated Ontology Merging and Alignment. *Proceedings of AAAI-2000*, pages 450Ű455, Austin (TX US). MIT Press/AAAI Press, 2000.
[29] Natalya Noy and Mark Musen. Anchor-prompt: using non-local context for semantic matching. In *Proceedings of the workshop on Ontologies and Information Sharing at the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 63-70, 2001.
[30] Anhai Doan, Jayant Madhavan, Robin Dhamankar, Pedro Domingos and, Alon Halevy. Learning to match ontologies on the Semantic Web. *The VLDB Journal*, **12**:303-319, 2003.
[31] Marc Ehrig and Steffen Staab. QOM - Quick Ontology Mapping. *Proceedings of 3rd ISWC*, Hiroshima (JP), pages 683-697, Springer-Verlag, 2004.
[32] Ontology from Russia. http://www.aifb.uni-karlsruhe.de/WBS/meh/foam/ontologies/russia1.owl. Last visit: 24-jul-2008.
[33] Ontology from Russia. http://www.aifb.uni-karlsruhe.de/WBS/meh/foam/ontologies/russia2.owl. Last visit: 24-jul-2008.
[34] Yi Li, Juan-Zi Li, Duo Zhang, Jie Tang: Result of Ontology Alignment with RiMOM at OAEI'06. Ontology Matching 2006.
[35] Frank van Harmelen: Two Obvious Intuitions: Ontology-Mapping Needs Background Knowledge and Approximation. IAT 2006: 11.
[36] Fausto Giunchiglia, Pavel Shvaiko, Mikalai Yatskevich: Discovering Missing Background Knowledge in Ontology Matching. ECAI 2006: 382-386.
[37] Ruben Vazquez, Nik Swoboda: Combining the Semantic Web with the Web as Background Knowledge for Ontology Mapping. OTM Conferences (1) 2007: 814-831.
[38] Rudi Cilibrasi, Paul M. B. Vitányi: The Google Similarity Distance. IEEE Trans. Knowl. Data Eng. 19(3): 370-383 (2007).
[39] Risto Gligorov, Warner ten Kate, Zharko Aleksovski, Frank van Harmelen: Using Google distance to weight approximate ontology matches. WWW 2007: 767-776.
[40] Marco Ernandes, Giovanni Angelini, Marco Gori: WebCrow: A Web-Based System for Crossword Solving. AAAI 2005: 1412-1417.
[41] Bibliography vocabulary from Yale University. http://www.cs.yale.edu/ dvm/daml/bib-ont.daml. Visit date: 28-jul-2008.
[42] Bibliography vocabulary from IS Institute. http://www.isi.edu/webscripter/bibtex.o.daml. Visit date: 28-jul-2008.
[43] AKT Ontology. http://www.aktors.org/ontology/portal. Visit date: 28-jul-2008.
[44] Ontology for computer science academic departments. http://www.cs.umd.edu/projects/plus/DAML/onts/ cs1.0.daml. Visit date: 28-jul-2008.
[45] Ontology about a subset of the GEDCOM data model. http://orlando.drc.com/daml/Ontology/Genealogy/ current/. Visit date: 28-jul-2008.
[46] Ontology about the GEDCOM data model. http://www.daml.org/2001/01/gedcom/gedcom. Visit date: 28-jul-2008.
[47] Carmel Domshlak, Avigdor Gal, Haggai Roitman: Rank Aggregation for Automatic Schema Matching. IEEE Trans. Knowl. Data Eng. 19(4): 538-553 (2007).
[48] Qiu Ji, Weiru Liu, Guilin Qi, David A. Bell: LCS: A Linguistic Combination System for Ontology Matching. KSEM 2006: 176-189.
[49] Stephanie Forrest: Genetic Algorithms. The Computer Science and Engineering Handbook 1997: 557-571.
[50] David Urdiales-Nieto, Jorge Martinez-Gil, Jose F. Aldana-Montes: MaSiMe: A Customized Similarity Measure and Its Application for Tag Cloud Refactoring. OTM Workshops 2009: 937-946.
[51] J. Wang, Z. Ding, C. Jiang: GAOM: Genetic Algorithm based Ontology Matching. In Proceedings of APSCC, 2006.
[52] Gerard D. Cohen, Simon Litsyn, Gilles Zémor: On greedy algorithms in coding theory. IEEE Transactions on Information Theory 42(6): 2053-2057 (1996).
[53] Yoon. Lee, Mayssam Sayyadian, AnHai Doan, Arnon Rosenthal: eTuner: tuning schema matching software using synthetic scenarios. VLDB J. 16(1): 97-122 (2007).
[54] Pat Langley: Elements of Machine Learning. 1994, ISBN 1-55860-301-8.
[55] Marc Ehrig, Steffen Staab, York Sure: Bootstrapping Ontology Alignment Methods with APFEL. International Semantic Web Conference 2005: 186-200.
[56] Michael I. Jordan, Christopher M. Bishop: Neural Networks. The Computer Science and Engineering Handbook 1997: 536-556.
[57] Jingshan Huang, Jiangbo Dang, José M. Vidal, and Michael N. Huhns. Ontology Matching Using an Artificial Neural Network to Learn Weights. IJCAI Workshop on Semantic Web for Collaborative Knowledge Acquisition 2007.
[58] Esko Ukkonen: Approximate String Matching with q-grams and Maximal Matches. Theor. Comput. Sci. 92(1): 191-211 (1992).
[59] Giorgos Stoilos, Giorgos B. Stamou, Stefanos D. Kollias: A String Metric for Ontology Alignment. International Semantic Web Conference 2005: 624-637
[60] Jorge Martinez-Gil, Ismael Navas-Delgado, Antonio Polo-Marquez, Jose F. Aldana-Montes: Comparison of Textual Renderings of Ontologies for Improving Their Alignment. CISIS 2008: 871-876.
[61] Ricardo A. Baeza-Yates, Berthier A. Ribeiro-Neto: Modern Information Retrieval. ACM Press / Addison-Wesley 1999, ISBN 0-201-39829-X.
[62] Michael K. Buckland, Fredric C. Gey: The Relationship between Recall and Precision. JASIS 45(1): 12-19 (1994).
[63] Gerd Stumme, Alexander Maedche: FCA-MERGE: Bottom-Up Merging of Ontologies. IJCAI 2001: 225-234.
[64] Fausto Giunchiglia, Pavel Shvaiko, Mikalai Yatskevich: S-Match: an Algorithm and an Implementation of Semantic Matching. ESWS 2004: 61-75.
[65] Wei Hu, Gong Cheng, Dongdong Zheng, Xinyu Zhong, Yuzhong Qu: The Results of Falcon-AO in the OAEI 2006 Campaign. Ontology Matching 2006.

[66] Slawomir Niedbala: OWL-CtxMatch in the OAEI 2006 Alignment Contest. Ontology Matching 2006.
[67] Patrick Lambrix, He Tan: A Tool for Evaluating Ontology Alignment Strategies. J. Data Semantics 8: 182-202 (2007).
[68] Gerard Salton, Chris Buckley: Improving retrieval performance by relevance feedback. JASIS 41(4):288-297 (1990).
[69] Alexandros Chortaras, Giorgos B. Stamou, Andreas Stafylopatis: Learning Ontology Alignments Using Recursive Neural Networks. ICANN (2) 2005: 811-816.
[70] Jingshan Huang, Jiangbo Dang, José M. Vidal, and Michael N. Huhns. Ontology Matching Using an Artificial Neural Network to Learn Weights. IJCAI Workshop on Semantic Web for Collaborative Knowledge Acquisition 2007.
[71] Malgorzata Mochol, Elena Paslaru Bontas Simperl: A High-Level Architecture of a Metadata-based Ontology Matching Framework. DEXA Workshops 2006: 354-358.