Knowledge and Information Systems

Evaluation of Two Heuristic Approaches to Solve the Ontology Meta-Matching Problem

Jorge Martinez-Gil and José F. Aldana-Montes

Department of Computer Languages and Computing Sciences, University of Málaga, Spain

Abstract. Nowadays many techniques and tools are available for addressing the ontology matching problem, however, the complex nature of this problem causes existing solutions to be unsatisfactory. This work aims to shed some light on a more flexible way of matching ontologies: Ontology Meta-Matching which is a set of techniques to configure optimum ontology matching functions. In this sense, we propose two approaches to automatically solve the Ontology Meta-Matching problem. The first one is called MaSiMe, which is based on a greedy strategy to compute efficiently the parameters which configure a composite matching algorithm. The second approach is called GOAL and is based on a Genetic Algorithm which scales better for a large number of atomic matching algorithms in the composite algorithm and is able to optimize the results of the matching process.

Keywords: Ontology Meta-Matching, Knowledge Management, Information Integration

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 using the web resources. Therefore, most authors consider the Web as an universal medium for data, information, and knowledge exchange (Berners-Lee et al, 2001).

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 fundamental so that appropriate knowledge exchange

Received Dec 12, 2008
Revised Jul 21, 2009
Accepted Nov 08, 2009

in this extension of the Web can be possible; it allows organizations to model their 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 modeling their own knowledge: (a) it is very difficult or expensive for many organizations to reach agreement about a common standard, and (b) those standards which do exist often do not fit in with the specific needs of all the participants.

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 (Kiefer et al, 2007). Moreover, dealing with natural language often leads to a significant error rate. This last statement is very important for those authors who do not easily allow interpretation of the intellectual nature of a given concept by a machine (Doerr, 2001). However, they admit that automatic methods are far cheaper and can detect relations of which humans are unaware. We are convinced that the future lies in the coordinated combination of intellectual and automatic methods. But in this work, we are going to try to find *customized similarity functions* (CSF) in order to obtain the best ontology 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, based on 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, based on the idea that a short distance means greater similarity. Distance measures usually satisfy the mathematical axioms of a metric.

In practice, 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 (Widdows, 2004). Similarity measures, however, give us an idea about the probability of compared objects being the same (Pfitzner et al, 2009), but without falling into the psychological objections of a metric. So from our point of view, working with similarity measures is more appropriate for detecting correspondences between different ontologies belonging to same domain. In this sense, Ontology Meta-Matching could be considered a technique that automatically selects the appropriate similarity measures and their associated weights in the case where similarity measures need to be composed. It tries to avoid the research work which depends heavily on human thought. Ontology Meta-Matching can be performed in several ways. We have designed one greedy and one genetic algorithm. The greedy algorithm allows us to obtain reasonable results with a low computational cost and, the advantages inherent to the second strategy are its great accuracy and scalability. The main contributions of this work are:

- An introduction to the problem of Ontology Meta-Matching.
- A greedy algorithm for solving the problem automatically and efficiently.
- A genetic algorithm for optimizing the solutions to the problem.
- An empirical evaluation of the proposed algorithms and a discussion of their advantages.

The remainder of this article is organized as follows. Section 2 describes the problem statement related to the Ontology Meta-Matching problem, describes the preliminary definitions and properties that are necessary for understanding our proposal and and compares our proposal to the other approaches that try to solve the problem. Section 3 discusses a greedy strategy and a way to effectively compute it. Section 4 describes a genetic strategy and a way to implement it. Section 5 shows the empirical data that we have obtained from some experiments, including results obtained from a standard benchmark for ontology matching. Section 6 includes the related work section which compares our work with other approaches. And finally, in Section 7 the conclusions are discussed.

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, an alignment $'A$ is returned::

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

Where A' is a set of mappings. A mapping[1] 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 (Ehrig, 2007). The entities that 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 it 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 necessary to compose these simple 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, usually called matchers. Matchers 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. **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 (Navarro, 2001) for a classical approach to the problem or (Woon and Wong, 2009) for a more modern point of view.

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

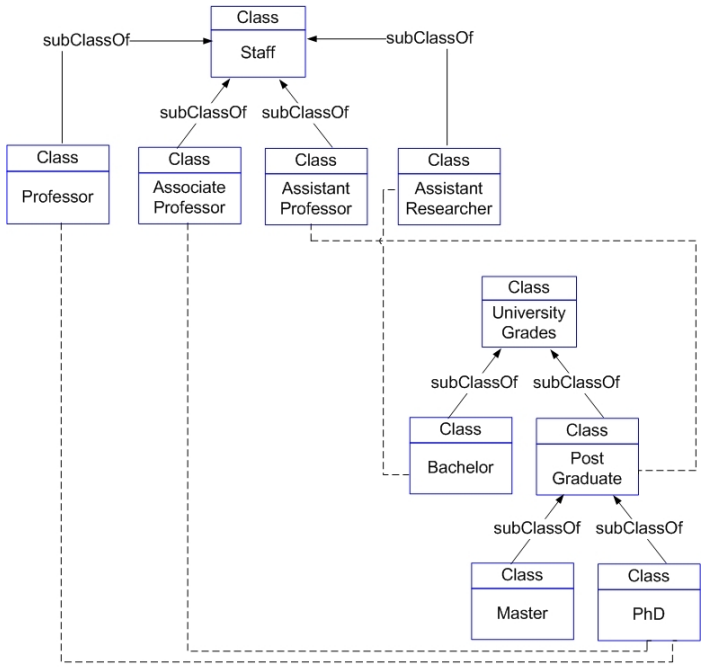


Fig. 1. Example of an user-dependent alignment. Most probably none of the two ontology owners will consider it optimal for them

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 involves the inclusion of linguistic resources such as lexicons and thesauri to identify possible similarities. Some of most popular linguistic method is to use WordNet (Wordnet, 2008) to identify some kinds of relationships between entities or background knowledge from the web in general (Rosenfeld and Feldman, 2009) or from more specific sources as Wikipedia as the work presented in (Wang et al, 2009).
5. **Inheritance analysis.** These 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 one. For example, if instances contain a string such as *years old*, it probably belongs to an attribute called *age*.
7. **Graph-Mapping.** This consists of 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. For example (Maguitman et al, 2006).
8. **Statistical analysis.** It consists of the extraction of keywords and textual de-

scriptions for detecting the meaning of the entities in relation to other entities. For example (Martinez-Gil et al, 2008).

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 (Euzenat and Shvaiko, 2007), semantic algorithms handle the input based on its semantic interpretation. One supposes that if two entities are the same, then they share the same interpretation. Thus, these are well grounded deductive methods. Most outstanding approaches are 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 in itself complicates the choice of the most appropriate for a given application domain. Secondly, 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 (Kiefer et al, 2007). In fact, due to effectively unlimited heterogeneity and the ambiguity of data description used in the ontology development, 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 necessary to use composite matchers. Using composite matchers has the advantage that all matching algorithms are considered as black boxes in order to select the most appropriates. In this way, automatically methods are able to tighten or loosen the matching function automatically if necessary. This means that it is not necessary for users to decide which algorithms have to be used.

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 area are COMA (Do and Rahm, 2002), COMA++ (Amueller et al, 2005), QuickMig (Drumm et al, 2007) and OntoBuilder (Gal et al, 2005), but these proposals 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, which is a set of ontologies that have been previously aligned by an expert.

2.1. 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 terms* μ_1 *and* μ_2 *to a similarity score* $sc \in \Re$ *in the range [0, 1].* This definition has been taken from (Kiefer et al, 2007)

Moreover, we need to know that a similarity score of 0 stands for complete inequality and 1 for equality of the input terms 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 1 (Weighted Similarity Measure). Let $\vec{A}$ be a vector 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 measure in the following form:

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

But from an 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 continuous and real interval) is infinite. For this reason, a brute force strategy would clearly be inefficient. It is necessary to look for better computational mechanisms that allow the problem of computing weighted 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 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 three real numbers stating the precision, recall and fall-out 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 value, thus, the similarity score closest to 1 that we can achieve.

2.2. Meta-matching Techniques

What exactly is ontology Meta-Matching in practice? *It is the technique of selecting the appropriate algorithms, weights and thresholds in ontology matching scenarios in order to obtain a satisfactory alignment between ontologies.* Figure 2 shows a diagram for modeling 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 accuracy and time consumption.

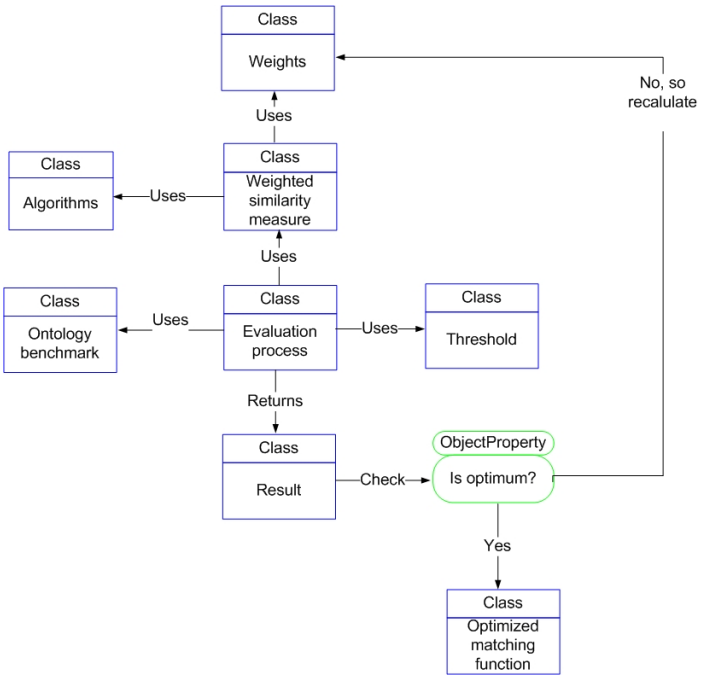


Fig. 2. General model for Meta-Matching

In general, we can describe the following as characteristics of a Meta-Matching task:

- It is not necessary for it to be done at runtime. Functions are computed and can be reused.
- It must be an automatic process, because it is a problem with a very complex computational nature.
- It must return the best possible matching function, thus, the function that best mimics the behavior of a human expert user.

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 (Domshlak et al, 2007) 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 entities. Where combinations can be as simple as: arithmetic or harmonic means, maximum, minimum, Minkowski distances, any kind of weighted product or sum, and so on, or more complex combinations like Linguistic Combinations (Ji et al, 2006)
- **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

(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 this case, it is said that parameters are optimized and, Greedy meta-matching, in such case, it is said that parameters are estimated. In Genetic Strategies, solutions tend to scale better and in Greedy techniques, solutions are obtained more quickly.
- **Based on Machine Learning meta-matching**, where the most outstanding approaches are Relevance Feedback and Neural networks training for meta-matching. In both cases, it is said that parameters are learned.

2.2.1. Heuristic Meta-Matching.

A heuristic is a method to help to solve a problem, commonly informal. It is used particularly 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 within a reasonable amount of time.

A lot of tools clearly implement heuristic Meta-Matching, we can see the clearest example in the default configuration of COMA (Do and Rahm, 2002), where an expert has initially adjusted the weights of the conceptual and structural techniques respectively. In order to avoid human interaction in this field, we can use Genetic Algorithms for optimizing the parameters or Greedy Algorithms to estimate them.

Based on Genetic Algorithm methods. Based on Genetic Algorithm methods (Forrest, 1997) are adaptive heuristic search algorithms premised on the evolutionary ideas of natural selection and genetics. The basic concept of GAs is designed to simulate the natural evolutionary system.

This approach is able to work with several goals (Martinez-Gil and Aldana-Montes, 2008): maximizing the precision, maximizing the recall, maximizing the fMeasure or reducing the number of false positives. Moreover, it has been tested combining some leading-edge similarity measures over a standard benchmark and produced some good results.

Another proposal is (Wang et al, 2006), a genetic algorithm-based optimization procedure for the ontology matching problem that is presented as a feature-matching process. First, from a global view, they model the problem of ontology matching as an optimization problem of a mapping between two compared ontologies, with each ontology having 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.

Greedy Meta-Matching Greedy (Cohen et al, 1996) 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 (Lee et al, 2001). Results from Greedy techniques are, in general, worse than those based on Genetics, but its computation time also tends to be much lower.

2.2.2. Based on Machine Learning Methods

Based on Machine Learning (Langey, 1994) Meta-Matching techniques considers both schema information and instance data. This kind of Meta-Matching can be divided into two subtypes[2] (i) Relevance Feedback and (ii) Neural Networks.

Relevance Feedback. This kind of approach explores the user validation of initial ontology alignments for automatically optimizing the configuration parameters of the matching strategies. A clear example of this kind of Meta-Matching is (Lee et al, 2001). Using such techniques we are able to avoid the user, and maybe the context, the dependency of the matching task, however, it implies spending a lot of time on training the systems. To do that automatically, it is possible to use Neural Networks.

Neural Networks Training. A neural network (Jordan and Bishop, 1997) 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 consists of training a neural network with heterogeneous enough benchmarks and then using the knowledge to predict new similarity functions. This is the case of (Huang et al, 2007) 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 (Chortaras et al, 2005).

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

3. 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 big O notation.

3.1. Maximum Similarity Measure

An initial 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 as we showed in Section 3, this kind of measure leads to an optimization problem for calculating the weight vector, because the number of candidates from the solution space is infinite. 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 polynomial time.

Definition 7. 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 2. 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 (Levenshtein, 1996), B = BlockDistance (Ziegler et al, 2006), M = MatchingCoefficient (Ziegler et al, 2006) , Q = QGramsDistance (Ukkonen, 1992)

This example shows an entity alignment, however, MaSiMe can be applied with different kinds of granularity:

- *Entity-Entity.* In this case, we may compute the best weights to align a pair of entities.
- *Ontology-Ontology.* In this case, MaSiMe may compute the best weighted function for aligning two input ontologies.
- *Benchmark Level.* This kind of calculation implies the computation of the best weights for a whole benchmark.

There are several properties for Definition 3:

Property 1 (Continuous uniform distribution). A priori, MaSiMe presents

a continuous uniform distribution in the interval $[0, 1]$, that is to say, its probability density function is characterized by

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

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 3. Let us suppose that we have: $\vec{A}$ = *(Google Similarity Distance (Ehrig, 2007), 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 matching algorithm for the equivalence of *(plane, aeroplane)* in this case would be:

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

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

Property 3 (Monotonocity). 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 (notation $\dot{}$) 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 4. Let us suppose we have the same conditions as in Example 3, 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 matching algorithm for the equivalence for *(plane, aeroplane)* is not the same as that in Example 3.

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

3.2. Computing the Weight Vector

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

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 following paragraphs discuss this mechanism.

3.2.1. 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.

Theorem 1 (About computing MaSiMe). *Let S be the set of all the matching algorithms, let A be the subset of S, thus, the set of matching algorithms that we want to use, let g be the granularity, let Q the set of positive Rational Numbers, let $i, j, k, ..., t$ be indexes belonging to the set of multiples for the granularity (denoted $\{\dot{g}\}$) then, a set of rational vectors r exists where each element r_i is result of the scalar product between A and the index pattern $(i, j-i, k-j, ..., 1-t)$. All of this subject to $j \geq i \wedge k \geq j \wedge 1 \geq k$. Moreover, the final result, called R, is the maximum of the elements r_i and is always less or equal than 1.*

And in mathematical form:

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

Proof (Theorem 1). *r_i is by definition the scalar product between a vector of matching algorithms that implements similarity measures and the pattern $(i, j-i, k-j, ..., 1-t)$. In this case, a similarity measure cannot be greater than 1 by Definition 1 and the sum of the pattern indexes cannot be greater than 1 by restriction $(i, j-i, k-j, ..., 1-t)$, so scalar product of such factors cannot be greater than 1.*

Now, we are going to show how to implement the computation of MaSiMe by using an imperative programming language. Algorithm 1 shows the pseudocode implementation for this theorem.

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

Input: ontology: O, O'
Input: algorithm vector: A
Input: granularity: g
Output: $MaSiMe$
foreach *pair* $(c1, c2)$ *of entities* $\in O$ *and* O' **do**
  foreach *index* $i, j, k, ..., t \in \kappa \times g$ **do**
    $result = A_1(c1, c2) \cdot i +$
    $A_2(c1, c2) \cdot j - i +$
    $A_3(c1, c2) \cdot k - j +$
    $A_4(c1, c2) \cdot t - k +$
    $...$
    $A_n(c1, c2) \cdot 1 - t$;
    if $result > MaSiMe$ **then**
      $MaSiMe = result;$
    end
    if $MaSiMe = 1$ **then**
      stop;
    end
  end
end

Algorithm 1*: The greedy algorithm to compute MaSiMe*

3.2.2. Complexity.

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

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

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

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

Example 5. Given the set $A = \{$*Levhenstein, BlockDistance, MatchingCoefficient, QGrams-Distance*$\}$, the complexity for the matching process using MaSiMe is calculated.

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

Brute force complexity for 4 matching algorithms may require 4 loops, MaSiMe requires 3 loops, so we can conclude that $O_m < O_{bf}$.

4. 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:

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

where x, y and z are weights to determine the importance of the three associated similarity measures, which belong, for instance, to the continuous interval [0, 1]. The problem that we want to solve is to find a good value of x, y and z in order 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 encoding in a string of values the contents of a tentative solution.
- Provide a numeric fitness function that will allow the relative quality of each individual tentative solution in a population to be rated.

4.1. GOAL

Now, we describe the method to solve meta-matching schemas that we have developed using Genetic Algorithms.

Definition 8. Genetics for Ontology ALignments (GOAL). *It is an elitist genetic algorithm which works under the paradigm of a single goal programming strategy. The idea behind GOAL is to automatically tune a combination of basic matching algorithms using sample cases solved by experts. The goal to optimize can be whatever specific aspect resulting from the evaluation of the matching tasks.*

GOAL is designed to work with lots of basic algorithms, all of them are chosen on the basis of their results, instead of their behavior, so can be considered as black boxes. The idea behind GOAL is that each component of the weight vector is associated to a basic matching algorithm. If we measure the distance between the result generated using the weight vector and the fitness value, we can know how good a weight vector is. In fact, we can know what the best weight vector is. Now we can interpret this vector; a value of 0 associated to a matching algorithm means that such algorithm is not relevant to solve the problem. A value between 0 and 1 means that the algorithm is relevant and if the value is 1 then the algorithm is able to solve the problem by itself.

Moreover, GOAL tries to not return local minimum for the weight vector associated to the matching function, because the crossover we introduce produces random chromosomes in order to extend the search space.

Apart from the quality of the results which we will evaluate in the next section, advantages inherent to GOAL are the great scalability and the possibility of avoiding the internal behavior of the basic matching algorithms.

4.2. Development

To develop GOAL, we have characterized the search space as some parameters. Then, we have encoded 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 have designed a fitness function to determine which chromosomes in the population are most likely to survive and reproduce using genetic crossover and mutation operations.

To return the fitness function, we can choose any parameter provided for the alignment evaluation process, i.e. precision, recall, fmeasure or fallout. In this way, we are providing the possibility of selecting one of these goals.

- Optimizing the precision ($fitness := precision$)
- Optimizing the recall ($fitness := recall$)
- Optimizing the fmeasure ($fitness := fmeasure$)
- Reducing the number of false positives ($fitness := fall - out$)

All of them are concepts used in Information Retrieval for measuring the quality 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). fmeasure is a weighted sum from precision and recall. Finally, false positives are relationships which have been provided to the user even though they are false (Buckland and Gey, 1994). In some domains, (for instance in medicine) false positives are absolutely unwanted. Take into account that we know all of them because we are using a set of test cases solved by an expert (or committee of experts). So the resulting matching function will behave ideally as experts who solve the sample cases.

4.3. Preliminary Study

We are going to do a preliminary 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 has 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 statistical distribution has 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 chosen a high value for the crossover between genes and, a small percentage for mutations, because we wish a classical configuration for the algorithm.
- After ten independent executions, we noticed that the genetic algorithm did not improve the results beyond the fifth generation, so we have set a limit of five generations.

4.4. Data to Repeat the Experiment

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

- As similarity measure vector:
  {*Levhenstein*(Levenshtein, 1996), *SIFO*(Martinez-Gil et al, 2008), *Stoilos*(Stoilos et al, 2005), *QGrams*(Ukkonen, 1992)}
- The GA has been configured taking the following parameters into account[3]:
  - 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 programming language was Java.

5. 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, there is a certain degree of agreement to use some measures from the Information Retrieval field (Baeza-Yates and Ribeiro-Neto, 1999). These are: *precision*, *recall*, *fmeasure* and *fallout*.

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

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

$$fmeasure = \frac{2 \times precision \times recall}{precision + recall}$$

$$fallout = \frac{\{non\ relevant\ mappings\} \cap \{retr.\ mappings\}}{\{non\ relevant\ mappings\}}$$

In Table 1, we explain briefly each of the test we are going to use to measure

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

Id	Brief explanation
101	Strictly identical ontologies
102	An ontology and a null ontology
103	An ontology and other with a lang. generalization
104	An ontology and other with a language restriction
201	Ontologies without entity names
202	Ontologies without entity comments
203	Ontologies without entity names and comments
204	Ontologies with different naming conventions
205	Ontologies whose labels are synonymous
206	Ontologies whose in different languages
221	An ontology and other with no specialization
222	An ontology and other with a flatenned hierarchy
223	An ontology and other with a expanded hierarchy
224	Identical ontologies without instances
225	Identical ontologies without restrictions
301	A real ontology about bibliography

Table 1. Brief explanation of the performed tests

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 misspelling	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 specialization	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 2. Behavior of MaSiMe for the standard benchmark of the OAEI

the quality of MaSiMe and GOAL using these formulas. These test cases belongs to the Ontology Matching Evaluation Initiative[4] (OAEI). These test cases try to measure the quality of methods when solving several use cases which are common in ontology matching scenarios. On the other hands, using this set of data means that GOAL will behave as international experts who solve the sample cases included in the benchmark.

Table 2 shows the results we have obtained for the greedy strategy called MaSiMe. Table 3 shows the results we have obtained for the genetic strategy called GOAL. Figure 3 shows a graphical comparison of the two strategies we have used. Next subsection discusses the results.

[4] http://www.ontologymatching.org

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 misspelling	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 specialization	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 3. Behavior of the GOAL for the standard benchmark of the OAEI

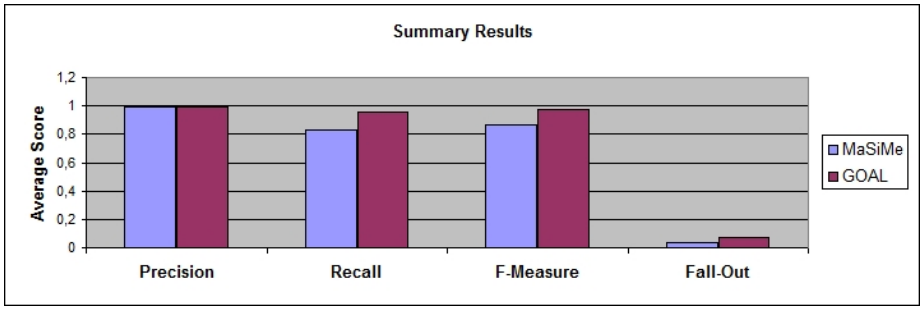


Fig. 3. Comparative results between strategies

5.1. Discussion

Precision takes all retrieved correspondences into account because measures the number of correct correspondences divided by the number of all returned correspondences. We reach an average precision of 99 percent because, most of times, it is possible to find a weighted algorithm combination which retrieve only correct correspondences.

On the other hand, in theory, it is easy to achieve recall of 100 percent by returning all possible correspondences between two ontologies. But recall alone is not enough; it is necessary to measure the number of non-relevant correspondences also. To to that, we compute the precision. So take into account that recall is computed in relation to the precision on all of our experiments.

So we have that getting only a good precision or only a good recall is not enough in general purpose scenarios. It is necessary to obtain a good combination of them. For this reason, fmeasure appeared. Remember that our strategy can improve whatever specific aspect of a matching task even fmeasure what it is really one of the great advantages of this work. That it is to say, we can improve the pair precision-recall when defining this fmeasure as the parameter to optimize. Possibility to optimize this parameter makes that GOAL is better in terms

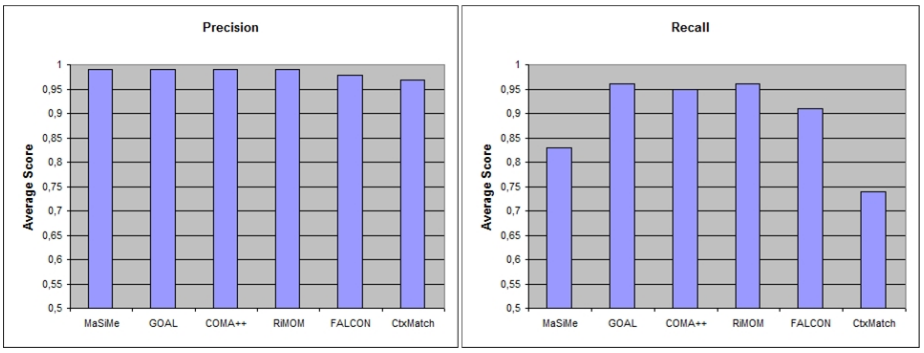


Fig. 4. Comparison with other tools

of accuracy than MaSiMe. MaSiMe is, however, an acceptable method for obtaining low cost results as we will see in the next section, where we compare these two strategies with other existing tools. In the future, we want to improve to a multiobjetive strategy (Pappa and Freitas, 2009) to avoid unwanted deviations from precision and recall values.

6. Related Work

If we look at the literature, we can distinguish between individual algorithms (i.e. Similarity Flooding (Melnik et al, 2002) or S-Match (Giunchiglia et al, 2004)) which apply 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 (i.e.Cupid (Madhavan et al, 2001)) follows a black box paradigm, in which various individual matchers are joined together in a new algorithm (Domshlak et al, 2007), while the so-called composite matchers allow increased user interaction (i.e. COMA++ (Amueller et al, 2005), Falcon (Hu et al, 2006), CtxMatch (Niedbala, 2006), RiMOM (Li et al, 2006)).

The problem is that those kinds of proposals, in the best of cases, use weights defined by an expert for configuring the matchers, but using our approaches involves computing the weights in an automatic way, so that the process can be more accurate. In Figure 4, we can see a comparison between some of the most popular tools for matching ontologies. The figure represents the arithmetic means of the values obtained for the standard benchmark for the precision and recall, obtaining the fmeasure and fallout is trivial.

To avoid the need for human expert intervention, there are three research lines now; one for evaluating the results of an alignment tool and maybe feedback the process (Gal et al, 2005) (Lambrix and Tan, 2007), another that tries to select automatically the algorithms according to their metadata (Mochol and Simperl, 2006) and another called Ontology Meta-Matching (Euzenat and Shvaiko, 2007) that tries to optimize automatically the parameters related to the 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:

6.1. Based on Exhaustive Search Solutions

Ontology Meta-Matching can be solved trivially by an exhaustive search when the number of similarity measures is low, one of the most outstanding approaches in this sense is eTuner (Lee et al, 2001) 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 large number of measures from a computational point of view. For this reason, most solutions try to avoid this kind of methods.

6.2. Based on Machine Learning Solutions

Based on Machine Learning Meta-Matching techniques can be divided into two subtypes: Relevance feedback (Salton and Buckley, 1990) and Neural Networks (Jordan and Bishop, 1997):

- The idea behind relevance feedback (Salton and Buckley, 1990) 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) (Ehrig et al, 2005) is a machine learning approach that explores user validation of initial alignments for optimizing automatically the configuration parameters of some of the matching strategies of the system, e.g., weights, thresholds, for the given matching task.
- Neural Networks (Jordan and Bishop, 1997) 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 of data. SFS (Huang et al, 2007) 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 so, it uses the neural networks technique.

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

6.3. Based on Genetic Algorithms Solutions

In relation to solutions based on Genetic Algorithm approaches, the most outstanding tool is Genetic Algorithms Based Ontology Matching (GAOM) (Wang et al, 2006) which is a genetic algorithm based approach for solving the ontology matching problem. For the purposes of better and more precise representation of ontology features, it defines the features from two points of view: intensional and extensional. On the other hand, the 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 4 shows a comparison of the results for both GAOM and our proposal.

Although we follow the same paradigm, it can be observed that GOAL shows

	precision	**recall**	**fmeasure**
GAOM	0.94	0.87	0.90
GOAL	**0.99**	**0.96**	**0.97**

Table 4. Comparison between GAOM and our proposal

slightly better results. We think that the main differences in relation to the other tool is that they do not use basic and widely tested matching solutions but their own algorithms and the fitness function that we use, because we do not force to the algorithm to find always a global solution, this fact makes worse the results. Therefore, as far as we know, our results constitute the new state of the art when solving meta-matching schemes using GAs.

7. Conclusions

We have presented Ontology Meta-Matching, 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 which depends heavily on human heuristics.

We have 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 environment. It is necessary 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 have designed and implemented two approaches to solve the Ontology Meta-Matching problem. Firstly, we have presented a greedy approach that is reasonably efficient, because is fast and reasonably accurate. Secondly, we have presented an approach based on Genetic Algorithms that is able to obtain more accurate results and is highly scalable, thus, it can be expanded with a lot of simple matching algorithms.

We have compared our two approaches with the most promising techniques and tools in the area of Ontology Meta-Matching. Like techniques, tools can be classified as either heuristic or learning-based. 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 results shows that our approach is in line with the best available tools and is able to overcome their limitations in some specific cases.

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

8. Acknowledgments

We wish to thank to all anonymous reviewers for their comments and suggestions which have helped to improve this work. We thank Enrique Alba for his comments in relation to the approach based on Genetic Algorithms. This work has been funded by the Spanish Ministry of Innovation and Science through the project: *ICARIA: A basic infrastructure for development in the Semantic Web and its application to conceptual mediation in bioinformatics.* TIN2008-04844 and *Pilot Project for Training and Developing Technologies for Applied Systems Biology*, P07-TIC-02978 (From the Regional Government of Andalucia).

References

Aumueller D, Do HH, Massmann S, Rahm E (2005) Schema and ontology matching with COMA++. In: SIGMOD Conference 2005, pp 906-908

Baeza-Yates R, Ribeiro-Neto BA (1999) Modern Information Retrieval. ACM Press / Addison-Wesley, ISBN 0-201-39829-X

Berners-Lee T, Hendler J, Lassila O (2001) The Semantic Web. Scientific American, May

Buckland MK, Gey FC (1994) The Relationship between Recall and Precision. JASIS 45(1): 12-19

Cabral L, Domingue J, Motta E, Payne TR, Hakimpour F (2004) Approaches to Semantic Web Services: an Overview and Comparisons. In ESWS 2004, pp 225-239

Chen H, Perich F, Finin TW, Joshi A (2004) SOUPA: Standard Ontology for Ubiquitous and Pervasive Applications. MobiQuitous, pp 258-267

Chortaras A, Stamou GB, Stafylopatis A (2005) Learning Ontology Alignments Using Recursive Neural Networks. In: ICANN (2) 2005, pp 811-816

Cilibrasi R, Vitanyi PMB (2007) The Google Similarity Distance. IEEE Trans Knowl Data Eng 19(3): 370-383

Gerard D. Cohen, Simon Litsyn, Gilles Zmor: On greedy algorithms in coding theory. IEEE Transactions on Information Theory 42(6): 2053-2057

Dietz JLG (2005) Enterprise Ontology. ICEIS 2005 (1): 5

Do HH, Rahm E (2002) COMA - A System for Flexible Combination of Schema Matching Approaches. In: VLDB 2002: pp 610-621

Doerr M (2001) Semantic Problems of Thesaurus Mapping. J. Dig. Inf. 1(8)

Domshlak C, Gal A, Roitman H (2007) Rank Aggregation for Automatic Schema Matching. IEEE Trans Knowl Data Eng 19(4): 538-553

Drumm C, Schmitt M, Do HH, Rahm E (2007) Quickmig: automatic schema matching for data migration projects. In CIKM 2007, pp 107-116

Ehrig M, Staab S, Sure Y (2005) Bootstrapping Ontology Alignment Methods with APFEL. In: International Semantic Web Conference 2005, pp 186-200

Ehrig M, Sure Y (2005) FOAM - Framework for Ontology Alignment and Mapping - Results of the Ontology Alignment Evaluation Initiative. Integrating Ontologies

Ehrig M (2007) Ontology Alignment: Bridging the Semantic Gap. (Contents), 2007 Springer, ISBN 978-0-387-36501-5

Euzenat J, Shvaiko P (2007) Ontology Matching. Springer-Verlag, 2007

Falconer S, Noy N (2007) Ontology Mapping - A User Survey. The Second International Workshop on Ontology Matching. In ISWC/ASWC, pp 49-60

Forrest S (1997) Genetic Algorithms. The Computer Science and Engineering Handbook 1997: pp 557-571

Gal A, Anaby-Tavor A, Trombetta A, Montesi D (2005) A framework for modeling and evaluating automatic semantic reconciliation. VLDB J. 14(1): 50-67

Giunchiglia F, Shvaiko P, Yatskevich M (2004) S-Match: an Algorithm and an Implementation of Semantic Matching. In: ESWS 2004, pp 61-75

He B, Chang KCC (2005) Making holistic schema matching robust: an ensemble approach. In: KDD 2005, pp 429-438

Hu W, Cheng G, Zheng D, Zhong X, Qu Y (2006) The Results of Falcon-AO in the OAEI 2006 Campaign. Ontology Matching

Huang J, Dang J, Vidal JM, Huhns MN (2007) Ontology Matching Using an Artificial Neural Network to Learn Weights. In: IJCAI Workshop on Semantic Web for Collaborative Knowledge Acquisition

Ji Q, Liu W, Qi G, Bell DA (2006) LCS: A Linguistic Combination System for Ontology Matching. In: KSEM 2006, pp 176-189

Jordan MI, Bishop CM (1997) Neural Networks. The Computer Science and Engineering Handbook, pp 536-556

Kiefer C, Bernstein A, Stocker M (2007) The Fundamentals of iSPARQL: A Virtual Triple Approach for Similarity-Based Semantic Web Tasks. In: ISWC/ASWC 2007, pp 295-309

Lambrix P, Tan H (2007) A Tool for Evaluating Ontology Alignment Strategies. J Data Semantics 8: 182-202

Langley P (1994) Elements of Machine Learning. 1994, ISBN 1-55860-301-8

Lee Y, Sayyadian M, Doan A, Rosenthal A (2001) eTuner: tuning schema matching software using synthetic scenarios. VLDB J. 16(1): 97-122

Levenshtein V (1966) Binary Codes Capable of Correcting Deletions, Insertions and Reversals. *Soviet Physics-Doklady*, Vol. 10, pp 707-710, August 1966

Li Y, Li JZ, Zhang D, Tang J (2006) Result of Ontology Alignment with RiMOM at OAEI'06. Ontology Matching

Maedche A, Motik B, Silva N, Volz R (2002) MAFRA - A MApping FRAmework for Distributed Ontologies. In EKAW 2002, pp 235-250

Madhavan J, Bernstein PA, Rahm E (2001) Generic Schema Matching with Cupid. In: VLDB 2001, pp 49-58

Maguitman A, Menczer F, Erdinc F, Roinestad H, Vespignani A (2006) Algorithmic computation and approximation of semantic similarity World Wide Web 9(4): 431-456

Martinez-Gil J, Alba E, Aldana-Montes JF (2008) Optimizing Ontology Alignments by Using Genetic Algorithms. Proceedings of the Workshop on Nature Based Reasoning for the Semantic Web. Karlsruhe, Germany.

Martinez-Gil J, Navas-Delgado I, Aldana-Montes JF (2008) SIFO. An efficient taxonomical matcher for ontology alignment. Technical Report ITI-08-3. Department of Languages and Computing Sciences, University of Malaga.

Martinez-Gil J, Navas-Delgado I, Polo-Marquez A, Aldana-Montes JF (2008) Comparison of textual renderings of ontologies for improving their alignment. Proceedings of the Second International Conference on Complex, Intelligent and Software Intensive Systems. Barcelona, Spain, pp 871-876

Melnik S, Garcia-Molina H, Rahm E (2002) Similarity Flooding: A Versatile Graph Matching Algorithm and Its Application to Schema Matching. Proceedings of International Conference on Data Engineering, pp 117-128

Mochol M, Bontas-Simperl EP (2006) A High-Level Architecture of a Metadata-based Ontology Matching Framework. In: DEXA Workshops, 354-358

Navarro G (2001) A guided tour to approximate string matching. ACM Comput Surv. 33(1): 31-88

Niedbala S (2006) OWL-CtxMatch in the OAEI 2006 Alignment Contest. Ontology Matching

Pappa GL, Freitas AA (2009) Evolving rule induction algorithms with multi-objective grammar-based genetic programming. Knowl and Inf Syst 19(3):283-309

Pfitzner D, Leibbrandt R, Powers D (2009) Characterization and evaluation of similarity measures for pairs of clusterings. Knowl and Inf Syst. 19(3):361-394

Roitman H, Gal A (2006) OntoBuilder: Fully Automatic Extraction and Consolidation of Ontologies from Web Sources Using Sequence Semantics. In: EDBT Workshops 2006, pp 573-576

Rosenfeld B, Feldman R (2009) Self-supervised relation extraction from the Web. Knowl Inf Syst. 17(1): 17-33

Salton G, Buckley C (1990) Improving retrieval performance by relevance feedback. JASIS 41(4):288-297

Stoilos G, Stamou GB, Kollias SD (2005) A String Metric for Ontology Alignment. In: Proceedings of International Semantic Web Conference 2005, pp 624-637

Kewei T, Yong Y: CMC: Combining Multiple Schema-Matching Strategies Based on Credibility Prediction. In: DASFAA 2005, pp 888-893

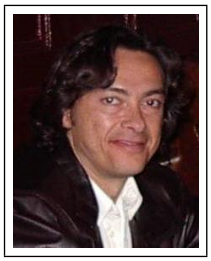
Ukkonen E (1992) Approximate String Matching with q-grams and Maximal Matches. Theor Comput Sci. 92(1): 191-211

Wang J, Ding Z, Jiang C (2006) GAOM: Genetic Algorithm based Ontology Matching. In: Proceedings of IEEE Asia-Pacific Conference on Services Computing, pp 888-893

Wang P, Hu J, Zeng HJ, Chen Z (2009) Using Wikipedia knowledge to improve text classification. Knowl Inf Syst. 19(3):265-281
Widdows D (2004) Geometry and Meaning. The University of Chicago Press, 2004
Woon WL, Wong KD (2009) String alignment for automated document versioning. Knowl Inf Syst. 18(3):293-309 (2009)
WordNet (2008). Available at `http://wordnet.princeton.edu`
Ziegler P, Kiefer C, Sturm C, Dittrich KR, Bernstein A (2006) Detecting Similarities in Ontologies with the SOQA-SimPack Toolkit. In: Proceedings of EDBT 2006, pp 59-76

Author Biographies

Jorge Martinez-Gil is currently Junior Researcher in the Department of Languages of Computing Sciences at the University of Malaga (Spain). He obtained his Bachelors and Master degree in Computer Science in the University of Extremadura at Caceres (Spain), and his MPhil in Software Engineering and Artificial Intelligence in the University of Malaga. His main research interests are related with the interoperability in the Semantic Web. In fact, his PhD thesis deals with the search for finding more flexible ways to match ontologies and other kinds of models to represent knowledge. Mr Martinez-Gil has published several papers in conferences and journals. He is a volunteer reviewer in some international journals related to his research field, and has been an invited researcher in some leading research groups.



José F. Aldana-Montes is currently professor in the Department of Languages of Computing Sciences at the Higher Computing School from the University of Malaga (Spain) and Head of Khaos Research, a group for researching about semantic aspects of databases. Dr Aldana-Montes has more than 20 years of experience in research about several aspects of databases, semistructured data and semantic technologies and its application to such fields as bioinformatics or tourism. He is author of several relevant papers in top bioinformatic journals and conferences. Related to teaching, he has been teaching theoretical and practical aspects of databases at all possible university levels: from undergraduate courses to PhD.

Correspondence and offprint requests to: José F. Aldana-Montes, Department of Languages and Computing Sciences, Boulevard Luis Pasteur s/n 29071 Málaga (Spain). Email: jfam@lcc.uma.es