The Knowledge Engineering Review, Vol. 00:0, 1–24. © 2010, Cambridge University Press
DOI: 10.1017/S0000000000000000 Printed in the United Kingdom

An Overview of Current Ontology Meta-Matching Solutions

JORGE MARTINEZ-GIL and JOSÉ F. ALDANA-MONTES

University of Málaga, Department of Computer Language and Computing Sciences
Boulevard Louis Pasteur 35, 29071 Málaga, Spain
E-mail: {jorgemar, jfam}@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 means that the existing solutions are unsatisfactory. This work intends to shed some light on a more flexible way of matching ontologies using ontology meta-matching. This emerging technique selects appropriate algorithms and their associated weights and thresholds in scenarios where accurate ontology matching is necessary. We think that an overview of the problem and an analysis of the existing state-of-the-art solutions will help researchers and practitioners to identify the most appropriate specific features and global strategies in order to build more accurate and dynamic systems following this paradigm.

1 Introduction

Most existing information systems use their own schemas to represent the data they handle. In order to obtain interoperability between agents, services or simply people that need to exchange information, correspondences must be established.

Nowadays ontologies are used to facilitate the exchange of information. Ontologies are formal representations of sets of concepts and relationships within a domain. But we are interested in the fact that ontologies are considered to extend the notion of schema. The reason is that an ontology can use more information than a traditional database schema, e.g., both hierarchical and non hierarchical information, as well as description information. Therefore, in comparison with classic schema matching, ontology matching has its own unique characteristics. Firstly, when comparing database schemas, ontologies provide greater flexibility and more explicit semantics for defining data. Secondly, database schemas are usually defined for specific databases, whereas an ontology aims to be reusable and sharable. Thirdly, ontology development is becoming a more and more decentralized procedure, although there are some exceptions such as the large-scale ontology SNOMED CT (Schulz *et al.*, 2007) which is developed centrally by only a few experts. Last but not least, ontologies have a larger number of representation primitives like cardinality constraints, inverse properties, transitive properties, disjoint classes, and type-checking constraints (Li *et al.*, 2009).

Therefore, the old problem of matching classic schemas has now evolved into an analog problem, although it is a little more complex. The task of finding correspondences between ontologies is called ontology matching and the output of this task is called ontology alignment (Euzenat & Shvaiko, 2007). In fact, obtaining satisfactory ontology alignments is a key aspect for such fields as:

- Semantic integration (Euzenat & Shvaiko, 2007). This is the process of combining metadata residing in 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 not for large volumes of information.

- Ontology mapping (Bernstein & Melnik, 2004). This 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. A typical use case for mapping is a query in one ontology representation, which is then rewritten and handed on to another ontology.
- The Web Services industry, where Semantic Web Services (SWS) are discovered and composed in a completely unsupervised manner. Originally SWS alignment was based on exact string matching of parameters, but nowadays researchers deal with heterogeneous and constrained data matching (Cabral *et al.*, 2004).
- Data Warehouse applications (He *et al.*, 2005). These kinds of applications are characterized by heterogeneous structural models that are analyzed and matched either manually or semi-automatically at design time. In such applications matching is a prerequisite of running the actual system.
- Similarity-based retrieval (Forbus *et al.*, 1995). Semantic similarity measures play an important role in information retrieval by providing the means to improve process recall and precision. These kinds of measures are used in various application domains, ranging from product comparison to job recruitment.
- Agent communication (Fasli, 2007). Existing software agents need to share a common terminology in order to facilitate the data interchange between them. Using ontologies is a promising technique to facilitate this process, but there are several problems related to the heterogeneity of the ontologies used by the agents which make the understanding at semantic level difficult. Ontology matching can solve this kind of problem.

All this means that business and scientific communities seek to develop automatic or semiautomatic techniques (known as matching algorithms or simply matchers) to reduce the tedious task of creating and maintaining the alignments manually. However, the nature of the problem is complex because "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). So we need mechanisms to make matching as independent as possible of data, context and users. A promising way of doing this is to combine similarity values predicted by multiple matchers to determine correspondences between ontology entities. In this way it will be possible to benefit from both the high degree of precision of some algorithms and at the same time the broader coverage of others (Eckert *et al.*, 2009). Ontology meta-matching tries to achieve this effectively and efficiently.

Although substantially different, this work complements the schema and ontology matching surveys presented in (Rahm & Bernstein, 2001), (Kalfoglou & Schorlemmer, 2003b), (Noy, 2004), (Shvaiko & Euzenat, 2005), and (Choi *et al.*, 2006) where ontology matching methods and tools are reviewed in detail, while the main contribution of this work is related to ontology meta-matching, thus, related to effective and efficient use of the techniques described in these surveys, which is one of the most important future challenges for semantic integration according to (Shvaiko & Euzenat, 2008). Whereas in our previous paper (Martinez-Gil & Aldana-Montes, 2009) we designed, implemented and evaluated two ontology meta-matching approaches, the current paper focusses on the following key points:

- An introduction to the notion of ontology meta-matching and its technical background.
- An analysis of the main techniques for ontology matching and their application to meta-matching.
- A qualitative explanation of the differences between some matcher combinations, matcher self-tuning and ontology meta-matching, terms that are often used inappropriately.
- An analysis of the existing state-of-the-art tools in this field.
- A discussion on the controversial issues concerning to meta-matching and the identification of the problems that remain open.

The rest of this work is organized as follows: Section 2 discusses the state-of-the-art related to ontology matching and why it is necessary to take into account mechanisms for exploiting simple ontology matchers. Section 3 describes the technical background necessary for understanding ontology meta-matching. Section 4 discusses the techniques that are used to meta-match. Section 5 presents an overview of the state-of-the-art tools on ontology meta-matching. Section 6 discusses the advantages and disadvantages of using meta-matching and, finally, in Section 7, we extract the conclusions from this work.

2 Problem Statement

At the present time, there are many thousands of ontologies available on the web (Martinez-Gil *et al.*, 2010). These ontologies are developed for different collections of information, and different kinds of applications. There are several reasons for the quick proliferation of ontologies, but we consider mainly two:

- It is often easier to construct a new ontology, than find an existing one which is appropriate for a given task.
- There is often a desire for direct control over the ontology for a particular domain, rather than having the structure dictated by external forces.

The main consequence of having large numbers of ontologies available is that we will have to integrate knowledge which is represented in different ways. Thus, in addition to the problem of integrating knowledge from different sources, we are now faced with the challenge of coping with different ontological representations of this knowledge. In relation to the first scenario, we require integrating the concepts of one ontology with another. This challenge is called the ontology matching problem and the key issue is the mapping of concepts and relationships from one ontology to another. Figure 1 shows an example of this scenario: there is an alignment between two ontologies representing landmarks and vehicles.

By examining two ontologies, it can be seen that ontology matching has to deal with the following five problems:

1. Concepts may have different names
2. Concepts may only be present in one or other of the ontologies
3. Concepts may be similar but not identical
4. Concepts may have similar notations but different semantics
5. There may be unstated assumptions in the ontology

On the other hand, the ontology matching problem could be reduced or avoided by adopting common ontologies. To this end, a number of efforts have been proposed with the intention of creating top-level ontologies, or definitive ontologies for a particular domain. An example of a top-level ontology is the IEEE Suggested Upper Merged Ontology (SUMO) (Oberle *et al.*, 2007) and examples of domain-specific ontologies include: the Gene ontology (Lomax, 2005), the OWL-Time ontology (Pan & Hobbs, 2005), and the Standard Ontology for Ubiquitous and Pervasive Applications (SOUPA) (Chen *et al.*, 2004).

However, people tend to match ontologies (Falconer & Noy, 2007) and, this is performed using (directly or indirectly) a six-step process that consists of the following steps proposed by Ehrig (Ehrig, 2006) and which are described below:

1. **Feature Engineering.** It consists of selecting excerpts of the overall ontology specification to describe a specific entity. Table 1 shows a complete list of ontology features that can be exploited by ontology matching techniques.
2. **Search Step Selection.** It consists of choosing two entities from the ontologies to compare them for an alignment.

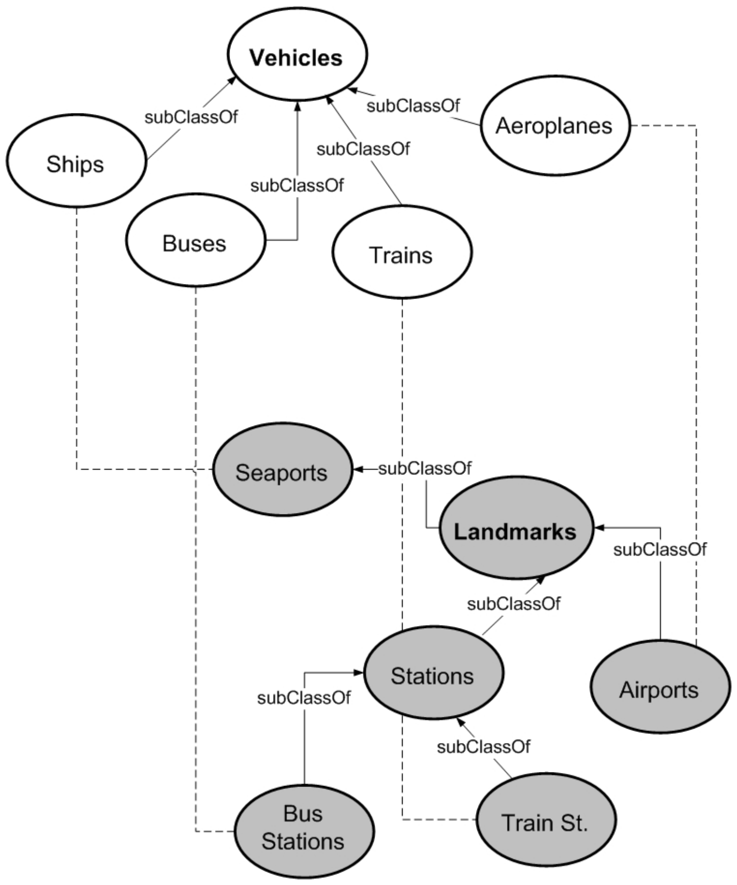

Figure 1 Example of alignment between two ontologies. Dotted lines indicate a kind of semantic correspondence between a landmark and kind of vehicle. The main goal of ontology matching is to solve this kind of situation automatically

3. **Matcher Assessment.** It consists of choosing a matching algorithm (matcher) for exploiting a given feature of the entities.
4. **Matcher Aggregation.** It consists of aggregating the multiple matcher for one entity pair into a single measure.
5. **Interpretation.** It consists of using all aggregated numbers, a threshold, and an interpretation strategy to decide whether two entities should eventually be aligned.
6. **Iteration.** The similarity of one entity pair influences the similarity of other entity pairs which are structurally connected to the first one, so the equality needs to be propagated through the ontologies.

The matchers from Step 2 can be linguistic matchers, structural matchers, constraint-based matchers or integration-knowledge-based matchers (depending on the feature to be exploited). It is also possible to create combinations of the matchers, in the attempt to overcome their limitations by proposing composite solutions. However, this is far from being a trivial task. Firstly, more and more matchers are constantly being developed, and this diversity by itself complicates the choice of the most appropriate one for a given application domain. Secondly, as one would expect, recent empirical analysis shows that there is no a single dominant matcher that performs best, regardless of the application domain. For this reason, it is necessary to introduce the notion of meta-matching.

Feature	Description
Linguistic Features	
Entity Name	The name of the ontology entity
Entity Documentation	Short textual description of the entity
Structural Features	
Entity Hierarchy	Information about the entity in the hierarchy
Relations	Relations of the entity to other entities
Attributes	Attributes of the entity
Constraint-based Features	
Data Type	The data type of the entity
Integration-Knowledge-based	
Technical Names	The technical annotation of the entity
Default Values	The default value (if applicable) for the entity
Identifiers	The local or global identifiers of the entity
Code Lists	Possible values for the attributes of the entity
Instance Data	Instances associated to the entity

Table 1 List of ontology features exploitable by ontology matching techniques

Feature to exploit	Algorithm's name
Linguistic Features	
Entity Name	Levenshtein Distance, Synonym similarity
Entity Documentation	Documentation Similarity (Tf-Idf)
Structural Features	
Entity Hierarchy	NamePath (3-Grams)
Relations	Children's name Algorithm (Base-2)
Attributes	Attribute's name Algorithm (Base-2)
Constraint-based Features	
Data Type	Trivial algorithm for comparing data types
Integration-Knowledge-based	
Technical Names	Google Distance
Default Values	Trivial algorithm for comparing default values
Identifiers	3-Grams
Code Lists	Wikipedia Distance
Instance Data	Instance-Based Algorithm (Dice)

Table 2 Example of matching algorithms that are categorized according to the techniques described in Table 1

2.1 Examples of Matchers

We show here several examples of well-known matchers and their associated explanation. Table 2 categorizes several existing matchers using the classification established in Table 1. Then, a more detailed description of the working mode for each algorithm is provided. The are two exceptions: matchers for comparing data types and default values have not been published in the past because they are trivial algorithms.

Levenshtein distance (Levenshtein, 1966). The Levenshtein distance between the names of two entities is given by the minimum number of operations needed to transform one name into other, where the operations can be insertions, deletions, or substitutions of a single character.

Synonym similarity (WordNet) (Pedersen *et al.*, 2004). Similarity measures quantify how similar two entities are, this decision is based on the information contained in an ISA-hierarchy. For this case, we are going to use the database called WordNet which organizes terms into hierarchies of ISA-relations.

Documentation Similarity (Tf-Idf). The documentation similarity algorithm uses the documentation of entities optionally available in some ontology languages. This algorithm assumes that two entities belonging to different ontologies are similar if their associated descriptions are also similar. Tf-Idf (Term frequency-Inverse document frequency)(Aizawa, 2003) has been chosen to compare these descriptions because it is an efficient algorithm for comparing short texts.

3-Grams. An n-gram is a subsequence of n tokens from a given name. The tokens can be phonemes, syllables, letters, and so on. The n-gram algorithm is used for efficient string matching. By converting a sequence of tokens to a set of n-grams (3 characters long in this case), it can be embedded in a vector space allowing the sequence to be compared to other sequences easily.

NamePath (3-Grams). The NamePath algorithm uses the complete path of an entity an ontology to calculate the similarity between entities. In an ontology, the Namepath is defined as the path from the ontology root to the given entity in this ontology.

Children's name Algorithm (Base-2). This technique is based on the detection of overlapping children's names from the entities to be compared. Base-2 means here that two entities are considered to represent the same object when two associated children overlap.

Attributes's name Algorithm (Base-2). This technique is quite similar to the detection of overlapping children's names from the entities to be compared. Base-2 means here that two entities are considered to represent the same real world object when two associated attributes are overlapped.

Google Distance (Cilibrasi & Vitanyi, 2007). The Google distance is a measure of semantic relatedness derived from the number of hits returned by the Google search engine for a given set of keywords. The idea behind this measure is that keywords with similar meanings in a natural language sense tend to be “close” in units of Google distance, while words with dissimilar meanings tend to be farther apart.

Wikipedia Distance. Wikipedia is the largest, free-content encyclopedia on the Internet. Wikipedia distance is similar to Google distance but we use Wikipedia content as corpus because it represents a great wealth of human knowledge. In order to use this content, the distance counts the hits returned by a Google search for the keywords after restricting the results with the option site:wikipedia.org

Instance-Based Algorithm (Relative overlap). This technique is based on the detection of overlapping instance's names from the entities to be compared. Relative overlap represents a value for the common instances. In the case of larger cardinality differences between instances, the relative overlap can be quite small.

3 Technical Background

In this section, we are going to define and explain the key concepts and examples that are necessary to understand the notion of ontology meta-matching.

Definition 1 (Similarity Measure). *A similarity measure sm is a function* $sm : \mu_1 \times \mu_2 \mapsto \Re$ *that associates the similarity of two input entities* μ_1 *and* μ_2 *to a similarity score* $sc \in \Re$ *in the range [0, 1].* This definition has been taken from (Ziegler *et al.*, 2006).

A similarity score of 0 stands for complete inequality and 1 for equality of the input solution mapping μ_1 and μ_2. Unlike a distance metric, where a similarity score of 0 stands for complete equality and a similarity score 1 is a bound to indicate complete inequality. Some authors think that a distance metric is not always appropriate because there are long-standing psychological objections to the axioms used to define it. 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). These objections are not relevant in this field because ontology matching is not directional. This means that, for example, the similarity between "car" and "automobile" is the same as between "automobile" and "car", but for normalization purposes we always consider 0 for inequality and 1 for equality.

On the other hand, we are interested in a special kind of similarity measures called configurable similarity measure. This kind of measure is a function that can be parametrized, thus, its behavior may vary depending on some external variable defined by an user. From the engineering point of view, configurable similarity measures share common characteristics: the search space is very large and the decision is made involving multiple criteria. Notice that resolving these simultaneously at run time makes the problem even harder. Example 1 shows a measure of this type called weighted similarity measure.

Example 1 (Weighted Similarity Measure). Let O_1 and O_2 be two ontologies. Let $\vec{w}$ a weight vector with $w_i \leq \kappa$, for some upper bound $\kappa \in R$. Let $\vec{A}$ be a vector of similarity measures. Then, the function wsf ranging over the unit interval $[0, 1] \subset R$ is defined as follows: $\mathsf{wsf}_{\vec{w},\vec{A}}(O_1, O_2) = max(\sum_{i=1}^{i=n} w_i \cdot A_i)$.

Example 1 shows a weighted similarity measure, thus, a function which leads to an optimization problem for calculating the weight vector, because the number of candidates from the solution space (in this case an arbitrary 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 2 (Ontology Matching). *An ontology matching om is a function* $om : O_1 \times O_2 \mapsto A$ *that associates two input ontologies* O_1 *and* O_2 *to an alignment A using a similarity measure.*

Now we define the output of an ontology matching function, i.e., an ontology alignment as follows.

Definition 3 (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 information purposes.*

It is also necessary to have methods that help us to distinguish between good and bad alignments. In fact, there are many ways to evaluate an ontology alignment:

- Compliance measures provide some insight on the quality of identified ontology 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 (Baeza-Yates & Ribeiro-Neto, 1999). These are precision and recall.

Definition 4 (Alignment Evaluation). *An alignment evaluation ae is a function* $ae : A \times A_R \mapsto precision \times recall$*, where precision and recall are real numbers ranging over the unit interval* $[0, 1]$. Precision states the fraction of retrieved correspondences that are relevant for a matching task. Recall is the fraction of the relevant correspondences that are obtained successfully in a matching task.

In this way, precision is a measure of exactness and recall a measure of completeness. Empirical studies of retrieval performance have shown a tendency for precision to decline as cecall increases. (Buckland & Gey, 1994) examined the nature of the relationship between precision and recall in more depth. On the other hand, in order to obtain a way to compare systems or techniques, an f-measure is defined as a weighting factor between precision and recall. The most common configuration consists of weighting precision and recall equally.

On the other hand, normally the goal of a matching function is to get the best value for the f-measure when evaluating the ontology alignment generated. But in some cases this may not be true, because the application being built needs a well-balanced pair of precision and recall measures instead of an optimized f-measure. In any case it is very difficult, even for an expert, to decide on the best way to customize a similarity measure and to implement it in the form of a matching function. This is because the number of variables and parameters to be considered is too large. On the other hand, we have that without tuning, ontology matching systems often fail to exploit specific characteristics. So research is focused on automatically configuring a function and avoiding the work which depends on a lot on human heuristics.

4 Ontology Meta-Matching

The expression Ontology Meta-Matching was introduced in (Euzenat & Shvaiko, 2007) for naming systems that try to configure automatically ontology matching functions. Although other approaches have been proposed, only several works have dared to give an explicit definition for ontology meta-matching; (Lee *et al.*, 2007) use the following definition: the method “to select the right matching components to execute, and to adjust the multiple knobs (e.g., threshold, coefficients, weights, etc.) of the components”. In (Martinez-Gil & Aldana-Montes, 2009), meta-matching is defined as “the technique of selecting the appropriate algorithms, weights and thresholds in ontology alignment scenarios”. The second definition does not include the selection of matching components because it assumes that all matchers are offered initially, and those which may be associated with a weight of 0 will be automatically deselected.

In general, there are several characteristics common to all meta-matching strategies:

- It is not necessary for it to be done at runtime. Matching functions can be computed in the background and then applied at runtime.
- It must be an automatic process. So it must be possible for it to be implemented by means of a software tool.

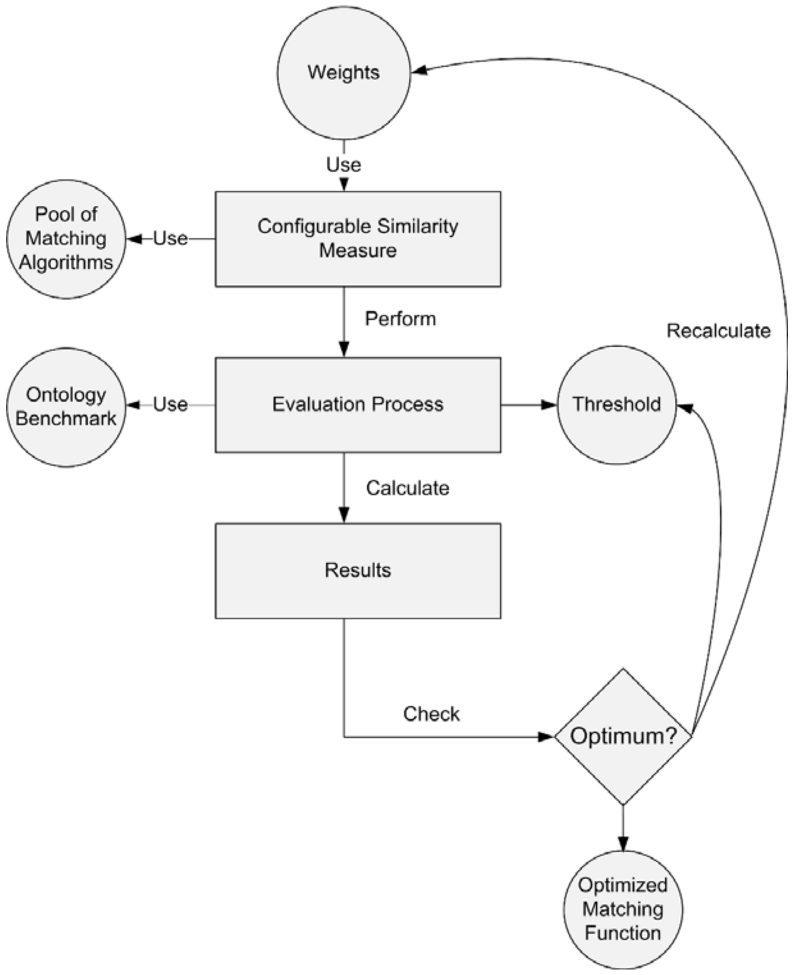

Figure 2 General model for meta-matching. Although some features can vary, most of the strategies present a solution schema similar to that presented here

- It must return the best possible matching function. If we do not know the best matching function, we can be satisfied with a function that behaves as a human expert would do.
- The evaluation of a meta-matching strategy is its returned matching function.

Moreover, Figure 2 shows a diagram for modeling the general actions in a meta-matching process. Although some features can vary, the process of meta-matching consists of adjusting in a smart way several parameters (algorithms, combination formula, and thresholds) in order to replicate the results of a heterogeneous set of solved cases. The optimized function is supposed to solve cases similar to the given ones.

4.1 Matcher combination, matcher self-tuning, and meta-matching

The expressions matcher combination, matcher self-tuning and meta-matching are often confused. Therefore, we are going to explain the differences between them.

- Matcher Combination. It involves the combination of individual matchers belonging to libraries of matchers. This increases the complexity of the matching problem as several matchers must be put together and combined appropriately. So far, only design time toolboxes allow to do this manually.

- Matcher Self-Tuning. Self-tuning is a term used widely in systems theory to name strategies that are able to maintain system stability and performance. In this way, Matcher Self-Tuning approaches attempt to tune and automatically adapt matching solutions to the settings in which an application operates. Decisions are usually taken at runtime. For example, systems can choose very fast matchers when receiving a very large ontology as an input or can automatically reduce a threshold when many correspondences are not found between ontologies.
- Ontology meta-matching consists of combining a set of heterogeneous ontology matchers in a smart way. It is not necessary to be performed at runtime. The main issue here is the automatic combination of matchers, the finding of the most appropriate values for their associated weights, the thresholds and in general, any parameter which may affect the results of an evaluation. The goal is try to balance the weaknesses and reinforce the strengths of the components. Unlike matcher self-tuning, the end goal of meta-matching is not to keep a system running effectively, but to get a very good ontology matching function.

On the other hand, ontology meta-matching can be seen from three points of view: (i) from the point of view of pre-match efforts and post-match efforts, (ii) from the point of view of the algorithmic techniques used to obtain the matching function and, (iii) from the point of view of the computer science paradigm that makes the meta-matching possible. All of them are discussed below.

4.2 Pre-match efforts and post-match efforts

From the point of view of pre-match efforts and post-match we have two kinds of meta-matching:

Pre-metamatching consists of feature selection, training of matchers, parameter configuration, and specification of auxiliary information. The main goal is that all this is done automatically. Pre-metamatching is not necessary to be performed at runtime.

Post-metamatching consists of identifying false positives and false negatives. The goal is to avoid human intervention, so it is necessary to use background knowledge strategies to automatically improve the results generated. Current research focuses on using web measures in order to find the relatedness between the entities that have been matched (Gracia & Mena, 2008). In this way, it is possible to check if the results are consistent.

4.3 Algorithmic techniques

From the point of view of the algorithmic techniques used to obtain the final ontology matching function, there are several strategies:

- Aggregation. This technique 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, weighted product or sum, and so on, or more complex combinations.
- *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 like (Ji *et al.*, 2006).

4.4 Paradigms to build meta-matching solutions

There are several ways to build ontology meta-matching systems. Figure 3 shows meta-matching techniques categorized according to the paradigm that makes the meta-matching possible, i.e., how the parameters are recalculated. It should be noted that this problem can be solved trivially

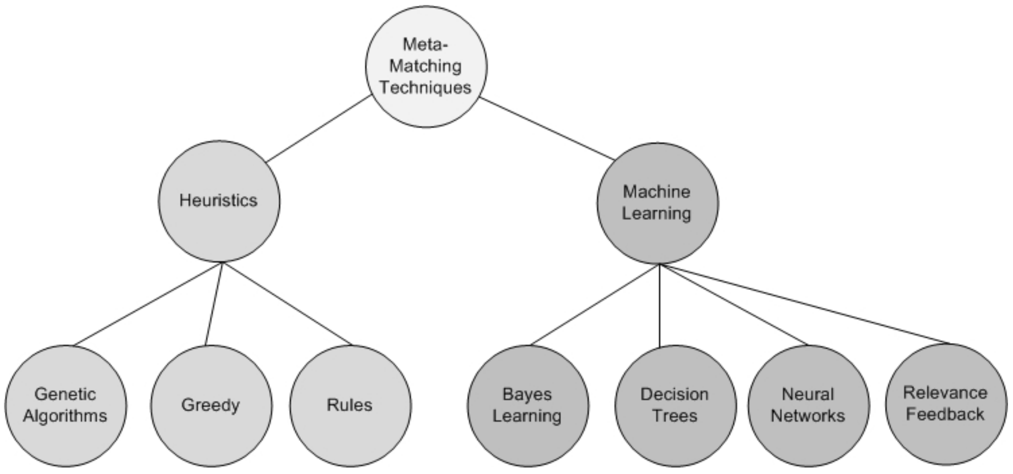

Figure 3 Existing ontology meta-matching techniques categorized according to the paradigm that makes it possible

by a brute force search when the number of matchers is low, but meta-matching scales better for a higher number of matchers. For this reason we do not include brute force methods as a viable technique. According to (Martinez-Gil & Aldana-Montes, 2009), these are the two main groups of techniques considered:

Heuristic meta-matching. 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; e.g., 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 this may not necessarily always be the case. 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.

1. *Based on Genetic Algorithms meta-matching.* Genetic Algorithms (GAs) (Forrest, 1997) are adaptive heuristic search algorithms based on the evolutionary ideas of natural selection. The basic concept of GAs is designed to simulate the natural evolutionary system.
2. *Greedy meta-matching.* Greedy meta-matching is a technique which, given a particular matching task, tries to automatically tune an ontology matching function. For this purpose, it tries to choose the best matchers and parameters to be used, but with a short-sighted strategy (Cohen *et al.*, 1996). Results from Greedy techniques are, in general, worse than those based on Genetics, although Greedy techniques also use much less computation time.
3. *Rules.* Rules are statements that define or constrain several aspects of the ontology matching tasks. eTuner (Lee *et al.*, 2007) uses perturbation rules in order to detect correspondences between source and target entities. The idea behind behind rules of this kind is to modify the source entity using well-known rules. If a perturbation coincides with the target entity then eTuner knows exactly what type of correspondence is the most appropriate to link the two entities.

Based on Machine Learning meta-matching, where the most outstanding approaches are Relevance Feedback, Bayes Learning, Decision Trees, and Neural networks training for meta-matching. In both cases, parameters are learned. Based on Machine Learning (Langley, 1994) meta-matching techniques considers both schema information and instance data. This kind of meta-matching can be divided into four subtypes.

1. *Relevance feedback.* This kind of approach explores the user validation of initial ontology alignments for automatically optimizing the configuration parameters of the matching strategies. Using such techniques we are able to avoid the user, and maybe the context, the dependency of the matching task, however, this approach involves spending much time on training the systems.
2. *Bayes learning for meta-matching.* Bayes learning is able to capture interdependencies among ontology matching algorithms and thus possibly improve the way they are combined. To do this, it proposes the use of Bayesian Networks (Svab & Svtek, 2008), a formal technique that can capture interdependencies among random variables. The main advantage of Bayesian networks compared to other uncertainty representation formalisms is that this approach allows rather complicated mutually related phenomena to be modelled quite easily.
3. *Decision trees.* A decision tree (or tree diagram) is a decision support tool that uses a tree-like graph or model of decisions and their possible consequences, including chance event outcomes, resource costs, and utility.
4. *Neural networks.* 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 benchmarks and then using the knowledge to predict new similarity functions.

4.5 Thresholds

When matching ontologies, a matcher can discard correspondences that do not reach some predefined threshold, assuming that those correspondences with low probability to be true are less appropriate than those with high probabilities. By doing so, a matcher increases precision, at the expense of recall. This means that it is necessary to take this parameter into account because a high value for it can reduce drastically (and not always wisely) the number of semantic correspondences discovered and vice versa. According to (Euzenat & Shvaiko, 2007), there are four main types of thresholds:

- **Hard threshold.** It selects only those values above a certain threshold value.
- **Delta threshold.** It is used as the threshold value of the greater similarity value, which is beyond a particular value given.
- **Proportional threshold.** It is used as the threshold percentage of the maximum value of the similarity values.
- **Percentage.** It retains the n% correspondences above the others.

Choosing appropriate thresholds is an important issue and it is not a trivial task. Indeed, manually choosing a proper configuration for the threshold is a very difficult process, even for an expert. Ontology meta-matching techniques can help so that this parameter can be optimized.

5 Existing Meta-Matching Tools

A number of tools have been developed which are based on the notion that humans are best equipped to perform ontology mappings. Thus, these tools are designed to present the task in such a way as to make it straightforward to define correspondences, e.g. by drawing links between concepts in a graphical interface. These software tools may also be combined with automated methods, which generate candidate matches for the user.

Another approach that we have taken into account is based on the definition of theoretical frameworks for ontology mapping. This is generally accomplished by considering the underlying Description Logics (DL) on which the ontologies are founded. Examples include formal concept analysis, information (IF-MAP) (Kalfoglou & Schorlemmer, 2003a).

A further possibility that exists is the manual combination of simple ontology matchers. Maybe the most popular tools to perform this task are: COMA (Hai Do & Rahm, 2002), COMA++ (Aumueller *et al.*, 2005), FOAM (Ehrig & Sure, 2005), and Ontobuilder (Roitman & Gal, 2006). All the tools belonging to this type of matching rely on human heuristic techniques.

Finally, we have considered two novel lines of research; one for evaluating the results of an alignment tool and to provide feedback to the process (Lambrix & Tan, 2007) and another, the object of this overview, called ontology meta-matching. It tries to optimize all the parameters related to the matching task. It should be taken into account that ontology meta-matching is not an end in itself, but a way to obtain high quality ontology alignments. Therefore, not all existing tools are pure meta-matching tools. MaSiMe (Martinez-Gil & Aldana-Montes, 2009), GAOM (Wang *et al.*, 2006), GOAL (Martinez-Gil*et al.*, 2008), eTuner (Lee *et al.*, 2007), APFEL (Ehrig *et al.*, 2005), MatchPlanner (Duchateau *et al.*, 2008), and YAM (Duchateau *et al.*, 2009) could be considered pure tools, while other tools are considered because they implement ontology meta-matching in any of the steps which they follow to solve problems. It should also be taken into account that several tools like Automatch (George Mason University)(Berlin & Motro, 2002), GLUE (University of Washington)(Doan *et al.*, 2003), SemInt (C&C/MITRE Corporation/Oracle)(Li & Clifton, 2000), and Rank Aggregation (Cornell University/Israel Institute of Technology)(Domshlak *et al.*, 2007) can only process classic schemas, and will therefore not be considered in this in this overview. The most outstanding tools in the area of heuristic ontology meta-matching are the following:

- **eTuner** (University of Illinois/MITRE Corporation), (Lee *et al.*, 2007). This is a meta-matching tool which, given a particular matching task, automatically tunes an existing ontology matching system (computing one-to-one alignments). For this purpose, it uses a perturbation rules algorithm that chooses the best parameters and the most effective basic matchers to be used.
- **GAOM** (Tongji University), (Wang *et al.*, 2006). GAOM (Genetic Algorithm based Ontology Matching) is a genetic algorithm-based optimization procedure for ontology matching problems which are presented as a feature-matching process. Firstly, the authors of this approach model the ontology matching problem as the optimization of a mapping between two compared ontologies. Each ontology has its associated feature sets. Secondly, as a heuristic search strategy, a genetic algorithm is used.
- **GOAL** (University of Malaga), (Martinez-Gil & Aldana-Montes, 2009). The GOAL approach is a genetic algorithm that is able to work with several goals: 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 such advantages as it is possible to optimize whatever specific aspect (precision, recall, fall-out or false positives)
- **MaSiMe** (University of Malaga), (Martinez-Gil & Aldana-Montes, 2009). MaSiMe is a customizable similarity measure that tries to establish the most appropriate weights for configuring a composite matching function to solve a set of matching cases. MaSiMe uses a greedy strategy that consists of using multiples from a value (called granularity) to reduce the solution space substantially. Then an exhaustive search is done over the reduced solution space.

Below are the most outstanding tools in the area of machine learning ontology meta-matching:

- **APFEL** (University of Karlsruhe/University of Koblenz Landau), (Ehrig *et al.*, 2005). APFEL means apple in German but it is here the acronym for Alignment Process Feature Estimation and Learning. In this work users are first given the ontology alignments for validation. These alignments are generated using previous proposals of the same authors. Using user validation, new hypotheses are generated by APFEL and weighted using the initial feedback.

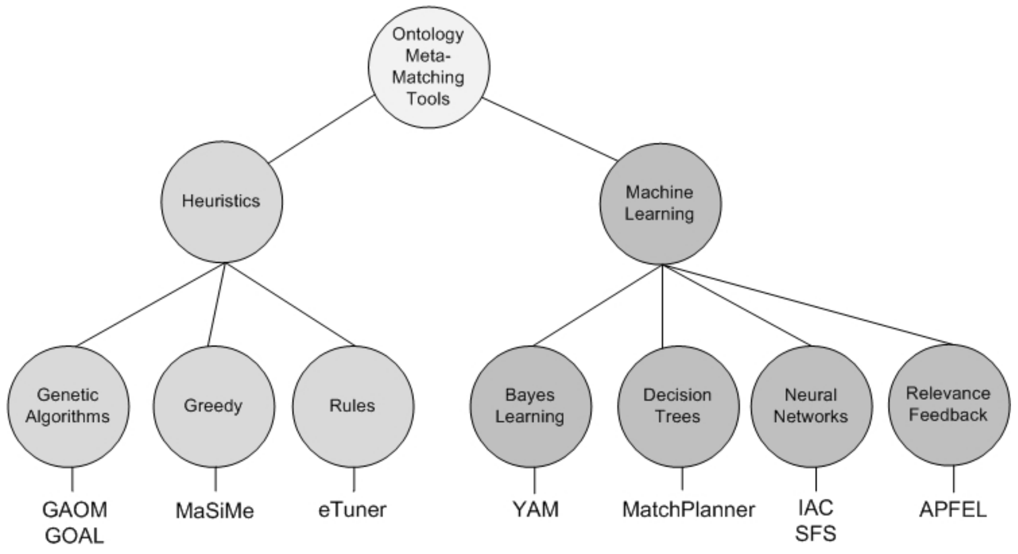

Figure 4 Existing meta-matching tools categorized according to the paradigm that makes them possible

- **IAC Neural Network** (SAP Research/Yahoo!/University of Pittsburgh) (Mao *et al.*, 2008). IAC stands for Interactive Activation and Competition. This proposal is based on a neural network to search for a global optimal solution that can satisfy as many ontology constraints as possible.
- **MatchPlanner** (University of Montpellier) (Duchateau *et al.*, 2008). It uses a decision tree to combine the most appropriate similarity measures for a given domain. In this way, the time performance of the tool is improved since the complexity is bounded by the height of the tree. This tool is also able to learn new decision trees, thus automatically tuning the tool to provide optimal configuration for a given matching scenario.
- **SFS** (University of South Carolina/Siemens Research) (Huang *et al.*, 2007). SFS (Super-concept Formation System) 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 an advanced neural network technique.
- **YAM** (University of Montpellier) (Duchateau *et al.*, 2009). According to the authors, YAM (Yet Another Matcher) is not (yet) another schema matching tool, as it enables the generation of a schema matcher according to user requirements. These requirements include a preference for recall or precision, a training data set and provided expert mappings. According to the authors, YAM uses a knowledge base that includes a set of similarity measures and classifiers. Based on the user requirements, YAM learns how to best apply these tools in concert to achieve quality for results.

Figure 4 shows the categorization of the listed ontology meta-matching tools according to the computational paradigm used to implement them. As can be seen, researchers have explored all the branches of the tree described in Section 4.

5.1 Comparison of the tools

In this subsection we compare the existing ontology meta-matching tools. It should be taken into account that it is very difficult to compare the existing tools. One reason is that researchers have not used the same characteristics when presenting their approaches. Also many of the studied tools are not under free licenses. Nevertheless, we will attempt to show the characteristics that can be extracted strictly from the articles where the tools were presented. The characteristics that we have chosen are explained below.

- *Results.* Researchers have not used the same benchmark datasets in order to offer their results. So we are going to study this characteristic according to a fuzzy classification that will be described in more detail later. The intention is not to offer a tool ranking, but rather to try to understand the scope of the different approaches.
- *Time consumption.* This characteristic is related to the time required to obtain results. We are not going to measure the time quantitatively, but we know that the time needed depends largely on the strategy used, e.g., a brute force strategy is very expensive in terms of time consumption, while more sophisticated strategies can reduce the time needed, for example, trying to avoid some regions from the solution space.
- *Training needed.* One of the disadvantages of ontology meta-matching tools is that a lot of effort for training them is required. This characteristic seems to be inherent to machine learning tools, but we are going to consider the time for optimizing heuristic tools too.
- *Ranking generation.* (Domshlak *et al.*, 2007) showed that it is also possible to build rankings for matching functions using meta-matching techniques. This is possible because several meta-matching techniques are able to identify a numerical fitness for each matching combination and, therefore, it is trivial to use this numerical fitness in order to generate a ranking of alternative approaches.
- *Human behaviour.* This characteristic tries to measure if it is possible to simulate the behaviour of the users who train the tools. As we commented earlier, a meta-matching tool should return the best possible matching function. If we do not know which one is the best matching function (and in this case), we can be satisfied with a function that behaves as a human expert would do.
- *Instance Data.* This feature tries to measure if the tools are able to deal with not only ontology classes, properties and relations, but with individuals too. It should be taken into account that work with individuals is far from being a trivial task because there is not much additional information in an ontology about entities of this kind.

We have established a fuzzy categorization with five kinds of values for each of these characteristics.
Very good. When the tool is one of the best tools for a given characteristic.
Good. When the meta-matching tool successfully implements a characteristic.
Regular. When it is possible to implement or configure a given characteristic easily.
Not so good. When the characteristic could be implemented but with serious changes.
Bad. When the tools do not implement this characteristic at all.

Heuristic tools for ontology meta-matching share several common characteristics, for example, it is easy for them to generate rankings of alternative approaches and they do not need too much time on training. But it is very difficult to simulate the user behaviour.

eTuner made a great contribution to the field of meta-matching when proposing perturbation rules in order to automatically configure matching functions. To the best of our knowledge, eTuner is the only tool which is able to tune existing tools. The main characteristics of eTuner are: eTuner results are very good, moreover it is possible to work with instance data, time consumption is not high, and training is not needed due to the perturbation rules that have been used. However, the tool can not simulate the behaviour of a specific user; it is only possible to simulate the behaviour of the perturbation rule designers.

GAOM was the first tool to propose the use of Genetic Algorithms for solving the ontology matching problem. However, results are far from the state-of-the-art. Time consumption depends of the user needs, so we have chosen an intermediate value. Training is needed, but a set of previously solved cases can be used, so we have chosen an intermediate value too. It uses a technique based on a numerical fitness, so it is very easy to GAOM to generate a ranking of approaches. Lastly, synthetic cases are used to optimize the strategy, so user behaviour cannot be simulated.

GOAL is the state-of-the-art tool using evolutionary computation. It is based on a genetic algorithm which scales very well for combining a large number of atomic matching algorithms and it is able to optimize the results of the matching process. Results for GOAL are comparable to those obtained in the state-of the-art, although this depends largely on the basic matchers to be composed. Time consumption is high because it is necessary to follow an evolutionary strategy to configure the matching function. Training is needed, but existing alignments can be reused. It is also possible to simulate the human behaviour and work with instance data.

MaSiMe is the only greedy tool among those studied. It tries to choose the best matchers and parameters to be used, but with a short-sighted strategy. The advantage of using this strategy is that it is possible to rapidly configure a matching function, but this configuration is not often the best. Therefore, the results are often not very good. Time consumption and training needed parameters are not high. Moreover, MaSiMe cannot simulate human behaviour, because a statistical method is used, but it can work with instance data. Lastly, MaSiMe associates a numerical fitness to each matching combination, so it is easy to generate a ranking of combinations.

Machine Learning tools for ontology meta-matching share several common characteristics, for example, all of them need a lot of training and it is very difficult to generate rankings of alternative approaches to solve a given scenario. On the contrary, it is a natural advantage for the tools of this kind the ability to simulate the human behaviour.

APFEL was the first tool which explored the user validation of initial alignments for optimizing alignment methods. For this reason, APFEL is considered the most representative tool for Relevance Feedback and is considered the classical tool for meta-matching in the field of machine learning. Relevance Feedback makes it possible to simulate the behavior of humans using the tool. It is possible to work with instance data and results are good according to the authors. For time consumption we have chosen a lower intermediate value. As negative points, a lot of training effort is needed and it is not possible to generate a ranking for different combinations.

MatchPlanner makes use of a decision tree to combine the most appropriate matching algorithms. As a first consequence of using the decision tree, the performance of the system is improved since the complexity is bounded by the height of the decision tree. Simulation of the user behaviour and results are very good. Time consumption can be optimized, so we have chosen an intermediate value for it. The tool needs a lot of training, and generating a ranking is not possible. MatchPlanner can deal with instance data.

IAC Neural Network tries to find a global optimal solution that best satisfies ontology constraints. The experimental results show the approach dramatically improves the performance of preliminary results in some common scenarios. Characteristics for IAC are: results and time consumption for IAC are reasonable. As with tools based on machine learning, much training is required, and ranking generation is not easy. Authors do not explain if the tool is able to deal with instance data, however, user behavior solving the cases to train the neural network can be simulated.

SFS tries to optimize a feature vector. During automated matching among ontologies, different semantic aspects, i.e., concept names, concept properties, and concept relationships, contribute in different degrees to the matching result. Therefore, a vector of weights are needed to be assigned to these aspects. SFS tries to obtain the best vector of weights. These characteristics are quite similar to the previous tool that we have considered. Maybe it is because the two tools use neural networks. Authors do not provide information about instance data in their paper. However, results according to the dataset they have used are good, and it is possible for the tool to behave as a human would.

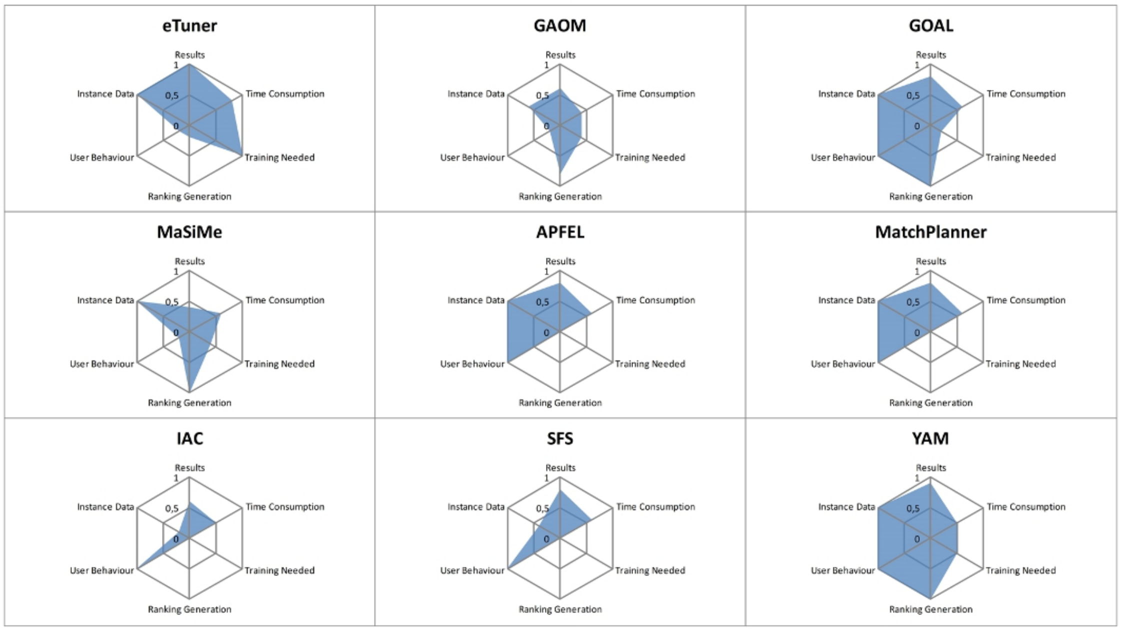

Figure 5 According to the characteristics that we have studied, we can conclude that the most complete tools, i.e. those tools with cover the greatest area in the radial graphics are eTuner and GOAL in the field of heuristic meta-matching, and APFEL and YAM in the field of machine learning

YAM is more than a meta-matching tool; it is a framework for researching meta-matching strategies. According to the authors, the idea of YAM is to train matchers with two errors which can occur when matching ontologies: discovering an irrelevant correspondence (false positive) and missing a relevant correspondence (false negative). The first error decreases precision while the second one decreases recall. Classifiers usually assign the same weight for both errors, but YAM tries to configure an optimal weight for them. On the other hand, YAM is possibly the most complete based-on-machine-learning tool for ontology meta-matching. It should be taken into account that it is the most modern too. Simulation of the user behaviour. The capability to work with instance data and the results presented are very good. Time consumption and training needs depend largely on the selected strategy so we have chosen an intermediate value for these two parameters. As a negative point, it is very difficult to offer a ranking of approaches using YAM.

Figure 5 shows a summary of the characteristics for the existing ontology meta-matching tools. We have chosen to represent the characteristics in a radial graphic. Each vertex of the graphic represents one of the proposed characteristics. In this way, it is easier to visualize the strengths and weaknesses for each ontology meta-matching tool. It should be taken into account that we are using the default configuration for each meta-matching tool, and we are not making a quantitative comparison, but a qualitative comparison.

6 Discussion

Using ontology meta-matching has several advantages over traditional techniques, but perhaps the most important one is that ontology meta-matching makes the matching independent from the data, context and the users, unlike the research work which depends a lot on human heuristics when tuning matching tools. Moreover, it is also possible to build rankings for the ontology matching functions as stated by (Domshlak*et al.*, 2007). This is possible because meta-matching techniques are able to identify the fitness for each matching combination. Moreover, it is possible to simulate the behaviour of the users who train the tools. However, there are also some controversial issues and unresolved problems too. The most important of them are: a) the problem of the ground truth, b) optimizing precision or recall is trivial, and c) meta-matching does not outperform blind benchmarks (yet).

6.1 The problem of the ground truth

Many critics believe that the meta-matching is a theoretical tool used to configure matching functions, but in practice, large sets of solved cases are necessary. The problem is that meta-matching systems are able to obtain good matching functions to face cases similar to the original cases, but these systems are not so good for dealing with new situations. This fact is commonly known as the problem of the ground truth. A partial solution is to optimize using a consensus of matchers, but this approach adds much noise to the process, so the quality of results is not very good.

6.2 Optimizing precision or recall is trivial

Many authors claim that their systems are able to optimize precision or recall. The problem here is that existing techniques can easily work in order to obtain high precision at the cost of the recall or, alternatively, recall can be increased at the cost of the precision. Several techniques try to optimize the f-measure, that it is to say, the equation that equally weights precision and recall in order to obtain a general measure, but the problem here is that current techniques are not able to avoid unwanted deviation between the precision and recall, they can only optimize a weighted equation. In this way, the best f-measure can be composed by a high precision and a low recall or vice versa. This problem could be solved by the application of multiobjective algorithms (Nebro *et al.*, 2008).

6.3 Meta-matching does not outperform blind benchmarks (yet)

Although meta-matching allows users to benefit both from the high degree of precision of some matchers and at the same time the broader coverage of others, in practice, researchers have not yet been able to find an appropriate implementation to outperform blind benchmarks. Outperforming well-known benchmarks has no practical implications, especially when these tools have been optimized for the benchmark. Meta-matching tools have been able to obtain results quite similar to best matching tools, but this still is not enough to justify their expensive development.

7 Conclusions

In this work, we have presented ontology meta-matching, as a novel computational discipline for fexible 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.

Secondly, we have provided an analysis of the most popular simple 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 process depends largely on the type of the underlying simple matchers used and the heterogeneity and soundness of the benchmarks for learning or optimizing the parameters.

Thirdly, we have provided a qualitative explanation of the differences between matcher combination, matcher self-tuning and ontology meta-matching which are terms that are often confused. Matcher combination is about obtaining complex matchers and is often performed manually, self-tuning is about improving accuracy and coverage at runtime, and meta-matching is about automatic matcher selection, combination and optimization.

We have shown the most promising tools in the area of ontology meta-matching. A lot of tools that clearly implement meta-matching have been developed recently. Like techniques, we have shown that tools can also be classified into heuristic or learning-based ones. Moreover, we have performed a study on these tools in order to identify their strengths and weaknesses. In order to facilitate tasks of this kind in the future, it would be a great idea for reseachers to use benchmarks like (Giunchiglia *et al.*, 2009) when presenting results.

Finally, we have discussed the advantages and disadvantages of using ontology meta-matching. We have explained throughout the work that meta-matching represents a serious effort to make the task of ontology matching a more independent process from users, context, and even the data involved, so we have focused the section for discussion on identifying explicitly the controversial issues and problems that remain open.

The lessons learned on ontology meta-matching will allow researchers to work with other kinds of conceptual schemas for modelling knowledge. In this sense, we are convinced that ontology meta-matching is a perfect candidate to take practitioners and users a step further in the state-of-the-art in terms of knowledge interoperability in increasingly demanding environments.

Acknowledgements

We wish to thank to all anonymous reviewers for their comments and suggestions which have helped to improve this work. This work has been funded by the Spanish Ministry of Innovation and Science through the project: ICARIA: From Semantic Web to Systems Biology, Project Code: TIN2008-04844 and by Regional Government of Andalucia through: Pilot Project for Training on Applied Systems Biology, Project Code: P07-TIC-02978.

References

Aizawa, A. 2003. An information-theoretic perspective of tf-idf measures. *Inf. Process. Manage.* 39(1) 45–65.

Aumueller, D., Hai Do, H., Massmann, S., Rahm, E. 2005. Schema and ontology matching with COMA++. *Proc. of SIGMOD Conference* 906–908.

Baeza-Yates, R., Ribeiro-Neto, B. 1999. *Modern information retrieval.* ACM Press / Addison-Wesley.

Berlin, J., Motro, A. 2002. Database schema matching using machine learning with feature selection. *Proc. of International Conference on Advanced Information Systems Engineering CAiSE'02* 452–466. Springer-Verlag.

Bernstein, P., Melnik, S. 2004. Meta Data management. *Proc. of International Conference on Data Engineering ICDE'04* 875. IEEE Computer Society.

Buckland, M., Gey, F. 1994. The relationship between recall and precision. *JASIS* 45(1) 12–19.

Cabral, L., Domingue, J., Motta, E., Payne, T., Hakimpour, F. 2004. Approaches to semantic web services: an overview and comparisons. *Proc. of European Semantic Web Conference ESWC'04* 225-239. Springer-Verlag.

Cilibrasi, R., Vitanyi, P. 2007. The Google similarity distance. *IEEE Trans. Knowl. Data Eng.* 19(3) 370–383.

Chen, H., Perich, F., Finin, T., Joshi, A. 2004. SOUPA: Standard Ontology for Ubiquitous and Pervasive Applications. *Proc. of Conference on Mobile and Ubiquitous Systems MobiQuitous'04* 258–267.

Choi, C., Song, I.,Han, H. 2006. A survey on ontology mapping. *ACM Sigmod Record* 35(3) 34–41.

Chortaras, A., Stamou, G., Stafylopatis, A. 2005. Learning ontology alignments using recursive neural networks. *Proc. of International Conference on Artificial Neural Networks ICANN'05* 2 811–816.

Doan, A., Madhavan, J., Dhamankar, R., Domingos, P., Halevy, A. 2003. Learning to match ontologies on the semantic web. *VLDB J.* 12(4) 303–319.

Cohen, D., Litsyn, S., Zemor, G. 1996. On greedy algorithms in coding theory. *IEEE Transactions on Information Theory* 42(6) 2053–2057.

Domshlak, C., Gal, A., Roitman, H. 2007. Rank aggregation for automatic schema matching. *IEEE Trans. Knowl. Data Eng. 19(4)* 538–553.

Duchateau, F., Coletta, R., Bellahsene, Z., Miller, RJ. 2009. (Not) yet another matcher. *Proc. of ACM Conference on Information and Knowledge Management CIKM'09* 1537–1540.

Duchateau, F., Bellahsene, Z., Coletta, R. 2008. A flexible approach for planning schema matching algorithms. *Proc. of On The Move Conferences (1) OTM'08* 249–264. Springer-Verlag.

Ehrig, M., Sure, Y. 2005. FOAM - Framework for Ontology Alignment and Mapping - Results of the ontology alignment evaluation initiative. *Proc. of Integrating Ontologies IO'05.*

Ehrig, M., Staab, S., Sure, Y. 2005. Bootstrapping ontology alignment methods with APFEL. *Proc. of International Semantic Web Conference ISWC'05* 186–200. Springer-Verlag.

Ehrig, M. 2006. Ontology alignment: Bridging the semantic gap. Springer-Verlag.

Eckert, K., Meilicke, C., Stuckenschmidt, H. 2009. Improving ontology matching using meta-level learning. *Proc. of European Semantic Web Conference ESWC'09* 158–172. Springer-Verlag.

Euzenat, J., Shvaiko, P. 2007. *Ontology matching.* Springer-Verlag.

Falconer, D., Noy, N. 2007. Ontology Mapping - An user survey. *Proc. of The Second International Workshop on Ontology Matching ISWC/ASWC'07* 49–60.

Fasli, M. 2007. On agent technology for e-commerce: trust, security and legal issues. *Knowledge Eng. Review* 22(1) 3–35.

Forbus, K., Gentner, D., Law, K. 1995. MAC/FAC: A model of similarity-based retrieval. *Cognitive Science* 19(2) 141–205.

Forrest, S. 1997. *Genetic algorithms. The computer science and engineering handbook.* 557–571.

Giunchiglia, F., Yatskevich, M., Avesani, P., Shvaiko, P. 2009. A large dataset for the evaluation of ontology matching. *Knowledge Eng. Review* 24(2) 137–157.

Gracia, J, Mena, E. 2008. Web-Based measure of semantic relatedness. *Proc. of Web Information Systems Engineering WISE'08* 136–150. Springer-Verlag.

Hai Do, H., Rahm, E. 2002. COMA - A system for flexible combination of schema matching approaches. *Proc. of Very Large Databases VLDB'02* 610–621.

He, B., Chen-Chuan Chang, K. 2005. Making holistic schema matching robust: an ensemble approach. *Proc. of Knowledge Discovery and Data Mining KDD'05* 429–438. Springer-Verlag.

Huang, J., Dang, J., Vidal, JM., Huhns, M. 2007. Ontology matching using an artificial neural network to learn weights. *Proc. of IJCAI Workshop on Semantic Web for Collaborative Knowledge.*

Ji, Q., Liu, W., Qi, G., Bell, D. 2006. LCS: A linguistic combination system for ontology matching. *Proc. of International Conference on Knowledge Science, Engineering and Management KSEM'06* 176–189.

Jordan, M., Bishop, C. 1997. Neural networks. *The Computer Science and Engineering Handbook.* 536–556.

Kalfoglou, Y., Schorlemmer, M. 2003. IF-Map: An ontology-mapping method based on information-flow theory. *J. Data Semantics* 1 98–127.

Kalfoglou, Y., Schorlemmer, M. 2003. Ontology mapping: the state of the art. *Knowledge Eng. Review* 18(1) 1–31.

Kiefer, C., Bernstein, A., Stocker, M. 2007. The fundamentals of iSPARQL: A virtual triple approach for similarity-based semantic web tasks. *Proc. of International/Asian Semantic Web Conference ISWC/ASWC'07* 295–309. Springer-Verlag.

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.* ISBN 1-55860-301-8.

Lee, Y., Sayyadian, M., Doan, A., Rosenthal, A. 2007. 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* 10 707–710.

Li, J., Tang, J., Li, Y., Luo, Q. 2009. RiMOM: A dynamic multistrategy ontology alignment framework. *IEEE Trans. Knowl. Data Eng.* 21(8) 1218–1232.

Li, WS., Clifton, C. 2000. SEMINT: A tool for identifying attribute correspondences in heterogeneous databases using neural networks. *Data Knowl. Eng.* 33(1) 49–84.

Lomax, J. 2005. Get ready to GO! A biologist's guide to the Gene Ontology. *Briefings in Bioinformatics* 6(3) 298–304.

Mao, M., Peng, Y., Spring, M. 2008. Neural network based constraint satisfaction in ontology mapping. *Proc. of Conference on Artificial Intelligence AAAI '08* 1207–1212. AAAI Press.

Martinez-Gil, J., Alba, E., Aldana-Montes, JF. 2008. Optimizing ontology alignments by using genetic algorithms. *Proc. of NatuReS.* CEUR-Proceedings.

Martinez-Gil, J., Aldana-Montes, J. 2009. Evaluation of two heuristic approaches to solve the ontology meta-matching problem *Knowledge and Information Systems*

Martinez-Gil, J., Alba, E., Aldana-Montes, J. 2010. Statistical Study about Existing OWL Ontologies from a Significant Sample as Previous Step for their Alignment. *Proc. of Conference on Complex, Intelligent and Software Intensive Systems CISIS*: 980–985. IEEE Computer Society.

Nebro, AJ., Luna, F., Alba, E., Dorronsoro, B., Durillo, JJ., Beham, A. 2008. AbYSS: Adapting scatter search to multiobjective optimization. *IEEE Trans. Evolutionary Computation* 12(4) 439–457.

Noy, N. 2004. Semantic integration: A survey of ontology-based approaches. *ACM Sigmod Record* 33(4) 65–70.

Oberle, D., Ankolekar, A., Hitzler, P., Cimiano, P., Sintek, M., Kiesel, M., Mougouie, B., Baumann, S., Vembu, S., Romanelli, M. 2007. DOLCE ergo SUMO: On foundational and domain models in the SmartWeb Integrated Ontology (SWIntO). *J. Web Sem.* 5(3) 156–174.

Pan, F., Hobbs, J. 2005. Temporal aggregates in OWL-Time. *Proc. of Florida Artificial Intelligence Research Society FLAIRS'05* 560–565.

Pedersen, T., Patwardhan, D., Michelizzi, J. 2004. WordNet::Similarity - measuring the relatedness of concepts. *Proc. of Ameriacan Asociation for Artificial Intelligence AAAI'04* 1024–1025. AAAI Press.

Rahm, E., Bernstein, P. 2001. A survey of approaches to automatic schema matching. *VLDB J.* 10(4) 334–350.

Roitman, H., Gal, A. 2006. OntoBuilder: Fully automatic extraction and consolidation of ontologies from web sources using sequence semantics. *Proc. of Extending Data Base Technology EDBT Workshops* 573–576. Springer-Verlag

Schulz, S., Suntisrivaraporn, B., Baader, F. 2007. SNOMED CT's Problem List: Ontologists' and Logicians' Therapy Suggestions. *MedInfo* 802–806.

Shvaiko, P., Euzenat, J. 2008. Ten challenges for ontology matching. *Proc. of On The Move Conferences OTM'08 (2)* 1164–1182. Springer-Verlag.

Shvaiko, P., Euzenat, J. 2005. A survey of schema-based matching approaches. *Journal on Data Semantics* 4 146–171.

Svab, O., Svtek, V. 2006. Ontology mapping enhanced using bayesian networks. *Proc. of Ontology Matching.*

Wang, J., Ding, J., Jiang, C. 2006. GAOM: Genetic Algorithm based Ontology Matching. *Proc. of Asia-Pacific Services Computing Conference APSCC'06.* IEEE Computer Society.

Widdows, D. 2004. *Geometry and meaning.* The University of Chicago Press.

Ziegler, P., Kiefer, C., Sturm, C., Dittrich, K., Bernstein, A. 2006. Detecting similarities in ontologies with the SOQA-SimPack toolkit. *Proc. of Extending Data Base Technology conference EDBT'06* 59–76. Springer-Verlag.