oMAP: Combining Classifiers for Aligning Automatically OWL Ontologies*

Umberto Straccia[1] and Raphaël Troncy[1,2]

[1] ISTI-CNR, Via G. Moruzzi 1, 56124 Pisa, Italy
{straccia, troncy}@isti.cnr.it
[2] CWI Amsterdam, P.O. Box 94079, 1090 GB Amsterdam, The Netherlands
raphael.troncy@cwi.nl

Abstract. This paper introduces a method and a tool for automatically aligning OWL ontologies, a crucial step for achieving the interoperability of heterogeneous systems in the Semantic Web. Different components are combined for finding suitable mapping candidates (together with their weights), and the set of rules with maximum matching probability is selected. Machine learning-based classifiers and a new classifier using the structure and the semantics of the OWL ontologies are proposed. Our method has been implemented and evaluated on an independent test set provided by an international ontology alignment contest. We provide the results of this evaluation with respect to the other competitors.

1 Introduction

The W3C recommendation of the *Resource Description Framework* (RDF) [22] and the *Web Ontology Language* (OWL) [19] languages is a new step towards the realization of the Semantic Web [4]. RDF aims to represent information and to exchange knowledge in the web, while OWL should be used to publish and share sets of terms called *ontologies*, supporting advanced web search, software agents and knowledge management. These languages are grounded on formal set-theoretic semantics, and specify meaning of concepts so that computers can process and understand them. They allow to infer new data from the knowledge already represented.

Ontologies are usually seen as a solution to data heterogeneity on the web [9]. An ontology is a way of describing the world: it allows to determine what kinds of things there are in the world, their characteristics, the relationships between them and more complex axioms [2]. Since a lot of efforts are deployed to provide hands-on support for developers of Semantic Web applications[1], with the online publishing of "best practices", it is expected now that more and more ontologies covering partially the same subjects will be available on the web. Indeed, this is already true for numerous complex domains such that the

* This work was carried out during the tenure of an ERCIM fellowship at CNR.
[1] W3C Semantic Web Best Practices and Deployment Working Group: http://www.w3.org/2001/sw/BestPractices/.

M. Kitsuregawa et al. (Eds.): WISE 2005, LNCS 3806, pp. 133–147, 2005.

medical [11] or the multimedia domain [14]. In such a case, some entities can be given different names or simply be defined in different ways or in different languages. The semantic interoperability has then to be grounded in ontology reconciliation. The underlying problem is often called the "ontology alignment" problem [9], that we address in this paper.

Comparing ontologies is useful for various tasks. During the building phase of the taxonomies, it is likely that the designer has to reuse some pieces of existing ontologies, internally developed or found on the web. Alignment methods are also necessary for dealing with the evolution and versioning issue of the ontologies (track changes, detect inconsistencies, merge, etc.). These methods can then be used for reformulating queries: documents annotated with respect to a *source* ontology can be retrieved even if the query uses terms from a *target* ontology. In the same way, documents classified under different web directories can be retrieved by comparing the heterogeneous web classes they belong to.

In this paper, we focus on ontologies described in the same knowledge representation language (OWL) and we propose a general framework combining several specific classifiers for aligning them automatically. We introduce a new classifier that uses the semantics of the OWL axioms for establishing equivalence and subsumption relationships between the classes and the properties defined in the ontologies.

The paper is organized as follows. We briefly present in the next section the OWL language as well as its syntax. Readers who are already familiar with this language can skip this section. Then, we introduce in section 3 *oMAP*, a framework whose goal is to find automatically the best mappings (together with their weights) between the entities defined in the OWL ontologies. The final mappings are obtained by combining the prediction of different classifiers. We describe the set of classifiers used: terminological, machine learning-based and we present a new one, based on the structure and the semantics of the OWL axioms. We sketch in section 4 the implementation of our framework. We have evaluated our method on an independent test set provided by an international ontology alignment contest and we show our results with respect to the other competitors in section 5. We present an overview of other alignment methods in section 6. Finally, we give our conclusions and outline future work in section 7.

2 Preliminaries: OWL Overview

OWL is a new formal language, recommended by the W3C, for representing ontologies in the Semantic Web [19]. In the Semantic Web vision [4], ontologies are used to provide structured vocabularies that explicate the relationships between different terms, allowing automated processes (and humans) to interpret their meaning flexibly yet unambiguously.

OWL has features from several families of representation languages, including primarily Description Logics [1] and frames. OWL can declare classes, and organize them in a subsumption ("subclass") hierarchy. OWL classes can be specified as logical combinations (intersections, unions, complements) of other

classes, or as enumerations of specified objects. In the same way, OWL can declare properties, organize them into a "subproperty" hierarchy, and provide domains and ranges for these properties. The domains of OWL properties are OWL classes while their ranges can be either OWL classes or externally-defined datatypes such as string or integer. OWL can state that a property is transitive, symmetric, functional or is the inverse of another property. OWL can express which objects (also called "individuals") belong to which classes, and what the property values are of specific individuals. Some axioms, such that the equivalence or disjointness between classes or the (in)equality between individuals, can be asserted. Finally, OWL is able to provide restrictions on how properties that are local to a class, behave. These restrictions may concern the type of all (or at least one) values of the property instances, or constrain their cardinality

Table 1. OWL Descriptions, Data Ranges, Properties, Individuals, and Data Values with examples (adapted from [12])

Abstract Syntax	DL Syntax	Example
Descriptions (C)		
A (URI reference)	A	`Conference`
`owl:Thing`	$\top$	
`owl:Nothing`	$\bot$	
`intersectionOf`(C_1 $C_2 \ldots$)	$C_1 \sqcap C_2$	`Reference` $\sqcap$ `Journal`
`unionOf`(C_1 $C_2 \ldots$)	$C_1 \sqcup C_2$	`Organization` $\sqcup$ `Institution`
`complementOf`(C)	$\neg C$	$\neg$ `MasterThesis`
`oneOf`(o_1 ...)	$\{o_1, \ldots\}$	{"WISE","ISWC",...}
`restriction`(R `someValuesFrom`(C))	$\exists R.C$	$\exists$ `parts.InCollection`
`restriction`(R `allValuesFrom`(C))	$\forall R.C$	$\forall$ `date.Date`
`restriction`(R `hasValue`(o))	$R : o$	`date : 2005`
`restriction`(R `minCardinality`(n))	$\geqslant n\ R$	$\geqslant 1$ `location`
`restriction`(R `maxCardinality`(n))	$\leqslant n\ R$	$\leqslant 1$ `publisher`
`restriction`(U `someValuesFrom`(D))	$\exists U.D$	$\exists$ `issue.integer`
`restriction`(U `allValuesFrom`(D))	$\forall U.D$	$\forall$ `name.string`
`restriction`(U `hasValue`(v))	$U : v$	`series : "LNCS"`
`restriction`(U `minCardinality`(n))	$\geqslant n\ U$	$\geqslant 1$ `title`
`restriction`(U `maxCardinality`(n))	$\leqslant n\ U$	$\leqslant 1$ `author`
Data Ranges (D)		
D (URI reference)	D	`string`
`oneOf`(v_1 ...)	$\{v_1, \ldots\}$	{"2004","2005",...}
Object Properties (R)		
R (URI reference)	R	`location`
Datatype Properties (U)		
U (URI reference)	U	`title`
Individuals (o)		
o (URI reference)	o	`WISE`
Data Values (v)		
v (RDF literal)	v	`"Int. Conf. SW"`

(i.e. they must be at least or at most a certain number of distinct values for this property) [12].

To illustrate this language, lets take a simple example on bibliographic data. A suitable bibliographic ontology, such as BibTeX, might represent the notions of book (e.g. monograph, proceedings) and part (e.g. chapter, article) that are reference entries; a journal or magazine should have a name, a publisher, a periodicity and contain some articles; a conference has a location and an organizer which is an institution that has an address, etc. The Table 1 summarizes the different constructs of the OWL language giving some examples within this domain. The first column gives the abstract syntax of OWL, the second, its equivalent in the Description Logics syntax, and the third some examples.

3 oMAP: A Framework for Automatically Aligning OWL Ontologies

This section introduces the *oMAP* framework for aligning automatically ontologies. Our approach is inspired by the data exchange problem [10] and borrows from others, like GLUE [6], the idea of combining several specialized components for finding the best set of mappings. The framework resumes partially the formalization proposed in [17] and extends the sPLMAP (*Schema Probabilistic Learning Mappings*) system to cope with the ontology alignment problem.

We draw in section 3.1 the general picture of our approach. Then, we detail several classifiers used to predict the *weight* of a possible mapping between two entities. These classifiers are terminological (section 3.2) or machine learning-based (section 3.3). Finally, we propose a classifier working on the structure and the formal semantics of the OWL constructs, thus using fully the meaning of the entities defined in the ontology (section 3.4).

3.1 Overall Strategy

Our goal is to automatically determine "similarity" relationships between classes and properties of two ontologies. For instance, given the ontologies in Figure 1, we would like to determine that an instance of the class `Conference` is likely an instance of the class `Congress`, that the property `creator` should subsume the property `author`, or that the class `Journal` is disjoint from the class `Directions`.

Theoretically, an ontology *mapping* is a tuple $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$, where $\mathbf{S}$ and $\mathbf{T}$ are respectively the source and target ontologies, and Σ is a finite set of *mapping constraints* of the form $\alpha_{i,j}\ T_j \leftarrow S_i$, where S_i and T_j are respectively the source and target entities. The intended meaning of this rule is that the entity S_i of the source ontology is mapped onto the entity T_j of the target ontology, and the confident measure associated with this mapping is $\alpha_{i,j}$. Note that a source entity may be mapped onto several target entities and conversely. But, we do not require that we have a mapping for every target entity.

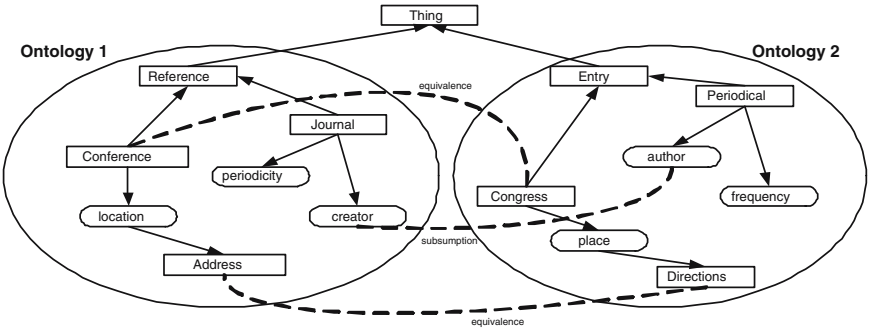

Fig. 1. Excerpt of two bibliographic ontologies and their mappings

Aligning two ontologies in *oMap* consists of three steps:

1. We form a possible set of mappings Σ, and estimate its quality based on the quality measures for its mapping rules;
2. For each mapping rule $T_j \leftarrow S_i$, we estimate its confidence $\alpha_{i,j}$, which also depends on the Σ it belongs to, that is, $\alpha_{i,j} = w(S_i, T_j, \Sigma)$;
3. As we cannot compute all possible sets Σ (there are exponentially many) and then choose the best one, we rather build iteratively our final set of mappings Σ using heuristics.

Similar to GLUE [6], we estimate the weight $w(S_i, T_j, \Sigma)$ of a mapping $T_j \leftarrow S_i$ by using different classifiers $CL_1, \ldots, CL_n$. Each classifier CL_k computes a weight $w(S_i, T_j, CL_k)$, which is the classifier's approximation of the rule $T_j \leftarrow S_i$. For each target entity T_j, CL_k provides a rank of the plausible source entities S_{i_k}. Then we rely on a priority list on the classifiers, $CL_1 \prec CL_2 \prec \ldots \prec CL_n$ and proceed as follows: for a given target entity T_j, select the top-ranked mapping of CL_1 if the weight is non-zero. Otherwise, select the top-ranked mapping provided by CL_2 if non-zero, and so on.

We present in the following several classifiers that have been used in our tests. It is worth noting that some of the classifiers consider the terminological part of the ontologies only, while others are based on their instances (i.e. the values of the individuals). Finally, we end this section by introducing a new classifier that fully uses the structure and the semantics of ontology definitions and axioms.

3.2 Terminological Classifiers

The terminological classifiers work on the name of the entities (class or property) defined in the ontologies. In OWL, each resource is identified by a URI, and can have some annotation properties attached. Among others, the `rdfs:label` property may be used to provide a human-readable version of a resource's name. Furthermore, multilingual labels are supported using the language tagging facility of RDF literals. In the following, we consider that the name of an entity is

given by the value of the `rdfs:label` property or by the URI fragment if this property is not specified.

Same entity names. This binary classifier CL_N returns a weight of 1 if and only if the two classes (or properties) have the same name, and 0 otherwise:

$$w(S_i, T_j, CL_N) = \begin{cases} 1 & \text{if } S_i, T_j \text{ have same name,} \\ 0 & \text{otherwise} \end{cases}$$

Same entity name stems. This binary classifier CL_S returns a weight of 1 if and only if the two classes (or properties) have the same *stem*[2] (we use the Porter stemming algorithm [20]), and 0 otherwise:

$$w(S_i, T_j, CL_S) = \begin{cases} 1 & \text{if } S_i, T_j \text{ have same } stem, \\ 0 & \text{otherwise} \end{cases}$$

String distance name. This classifier CL_{LD} computes some similarity measures between the entity names (once downcased) such that the Levenshtein distance [3] (or edit distance), which is given by the smallest number of insertions, deletions, and substitutions required to transform one string into the other. The prediction is then computed as:

$$w(S_i, T_j, CL_{LD}) = \frac{dist_{Levenshtein}(S_i, T_j)}{\max(length(S_i), length(T_j))}$$

3.3 Machine Learning-Based Classifier

As we have seen in section 2, an ontology often contains some individuals. It is then possible to use machine learning-based classifiers to predict the weight of a mapping between two entities. The instances of an OWL ontology can be gathered using the following rules: we consider (*i*) the label for the named individuals, (*ii*) the data value for the datatype properties and (*iii*) the type for the anonymous individuals and the range of the object properties.

For example, using the abstract syntax of [12], let us consider the following individuals :

```
Individual (x1 type (Conference)
        value (label "Int. Conf. on Web Information Systems Engineering")
        value (location x2))
Individual (x2 type (Address)
        value (city "New York city") value (country "USA"))
```

Then, the text gathered u_1 for the named individual x_1 will be (`"Int. Conf. on Web Information Systems Engineering", "Address"`) and u_2 for the anonymous individual x_2 (`"Address", "New York city", "USA"`).

[2] The root of the terms without its prefixes and suffixes.

[3] Levenshtein distance is named after the Russian scientist Vladimir Levenshtein, who devised the algorithm in 1965.

We describe in the following the typical and well-known classifier that we used in oMAP: the Naive Bayes [23].

Naive Bayes Text Classifier. The classifier CL_{NB} uses a Naive Bayes text classifier [23] for text content. Each class (or property) S_i acts as a category, and the instances are considered as bags of words (with normalized word frequencies as probability estimations). For each $(x, u) \in T_j$, the probability $Pr(S_i|u)$ that the value u should be mapped onto S_i is computed. In a second step, these probabilities are combined by:

$$w(S_i, T_j, CL_{NB}) = \sum_{(x,u)\in T_j} Pr(S_i|u) \cdot Pr(u)$$

Again, we consider the values as bags of words. With $Pr(S_i)$ we denote the probability that a randomly chosen value in $\bigcup_k S_k$ is a value in S_i. If we assume independence of the words in a value, then we obtain:

$$Pr(S_i|u) = Pr(u|S_i) \cdot \frac{Pr(S_i)}{Pr(u)} = \frac{Pr(S_i)}{Pr(u)} \cdot \prod_{m\in u} Pr(m|S_i)$$

Together, the final formula is:

$$w(S_i, T_j, CL_{NB}) = Pr(S_i) \cdot \sum_{(x,u)\in T_j} \prod_{m\in u} Pr(m|S_i)$$

If a word does not appear in the content for any individual in S_i ($Pr(m|S_i) = 0$), we assume a small value to avoid a product of zero.

3.4 A Structural and Semantics-Based Classifier

Besides these well-known algorithms in information retrieval and text classification, we introduce a new classifier, CL_{Sem}, which is able to use the semantics of the OWL definitions while being guided by their syntax. It is used in the framework *a posteriori*, in so far as the weighted average of the predictions of all the other classifiers will serve as its input. In the following, we note with $w'(S_i, T_j, \Sigma)$ the average weight of the mapping $T_j \leftarrow S_i$ estimated by the classifiers of the previous sections, where S_i (*resp.* T_j) is a concept or property name of the source (*resp.* target) ontology. Note that in case the structural classifier is used alone, we set: $w'(S_i, T_j, \Sigma) = 1$. The formal recursive definition of CL_{Sem} is then given by:

1. If S_i and T_j are property names:

$$w(S_i, T_j, \Sigma) = \begin{cases} 0 & \text{if } T_j \leftarrow S_i \notin \Sigma \\ w'(S_i, T_j, \Sigma) & \text{otherwise} \end{cases}$$

2. If S_i and T_j are concept names: let assume that their definitions are $S_i \sqsubseteq C_1 \dots C_m$ and $T_j \sqsubseteq D_1 \dots D_n$, and we note $\mathcal{D} = \mathcal{D}(S_i) \times \mathcal{D}(T_j)$[4], then:

[4] $\mathcal{D}(S_i)$ represents the set of concepts directly parents of S_i.

$$w(S_i,T_j,\Sigma) = \begin{cases} 0 & \text{if } T_j \leftarrow S_i \notin \Sigma \\ w'(S_i,T_j,\Sigma) & \text{if } |\mathcal{D}| = 0 \text{ and } T_j \leftarrow S_i \in \Sigma \\ \frac{1}{(|Set|+1)} \cdot \left(w'(S_i,T_j,\Sigma) + \max\limits_{Set} \left(\sum\limits_{(C_i,D_j)\in Set} w(C_i,D_j,\Sigma) \right) \right) & \text{otherwise} \end{cases}$$

3. Let $C_S = (QR.C)$ and $D_T = (Q'R'.D)$, where Q, Q' are quantifiers $\forall$ or $\exists$ or cardinality restrictions, R, R' are property names and C, D are concept expressions, then:

$$w(C_S, D_T, \Sigma) = w_Q(Q, Q') \cdot w(R, R', \Sigma) \cdot w(C, D, \Sigma)$$

4. Let $C_S = (op\ C_1 \dots C_m)$ and $D_T = (op'\ D_1 \dots D_m)$, where the concept constructors op, op' in the concepts C_S, D_T are in prefix notation, op, op' are the concept constructors among $\sqcap, \sqcup, \neg$ and $n, m \geq 1$, then:

$$w(C_S, D_T, \Sigma) = w_{op}(op, op') \cdot \frac{\max\limits_{Set} \left(\sum\limits_{(C_i,D_j)\in Set} w(C_i,D_j,\Sigma) \right)}{\min(m,n)}$$

where:

- $Set \subseteq \{C_1 \dots C_m\} \times \{D_1 \dots D_n\}$ and $|Set| = \min(m,n)$,
- $(C,D) \in Set, (C',D') \in Set \Rightarrow C \neq C', D \neq D'$.

We give in the Table 2 the values for w_Q and w_{op} we used.

Table 2. Possible values for w_{op} and w_Q weights

w_{op} is given by:

	$\sqcap$	$\sqcup$	$\neg$
$\sqcap$	1	1/4	0
$\sqcup$		1	0
$\neg$			1

w_Q is given by:

	$\exists$	$\forall$
$\exists$	1	1/4
$\forall$		1

	$\leq n$	$\geq n$
$\leq m$	1	1/3
$\geq m$		1

4 Implementation

Like we have seen in the section 3.1, our approach begins to form some possible Σ sets, for evaluating the weight of each mapping rules they contain. The generation of *all* possible Σ sets becomes quickly a critical issue since this number can be huge. We address this problem in the section 4.1. The *oMAP* framework allows to align any OWL ontologies, represented in the RDF/XML syntax. Hence, it uses extensively the OWL API [3] and the Alignment API available in JAVA (section 4.2).

4.1 Reduction of the Space

Lets assume that the two ontologies to be aligned have the following characteristics: a source ontology $\mathcal{S}$ containing C_S classes, OP_S object properties and DP_S

datatype properties; and a target ontology $\mathcal{T}$ containing C_T classes, OP_T object properties and DP_T datatype properties. Theoretically, the number of all possible mapping rules is given by: $Nb\_rules = (C_S \cdot C_T) + (OP_S \cdot OP_T) + (DP_S \cdot DP_T)$. Then, the number of all possible Σ sets containing these mapping rules is given by: $Nb_\Sigma = 2^{Nb\_rules}$. To reduce the number of Σ sets to generate and then to evaluate, we make a first approximation: we consider only the Σ sets that contain exactly one mapping rule for each definition of classes, object properties and datatype properties of the source or the target ontology (see section 3.4 for the formal definition of these sets). The number of all possible Σ sets is then given by:

$$Nb_\Sigma = \frac{Max(C_S, C_T)!}{|C_S - C_T|!} \cdot \frac{Max(OP_S, OP_T)!}{|OP_S - OP_T|!} \cdot \frac{Max(DP_S, DP_T)!}{|DP_S - DP_T|!}$$

We reduce also the number of Σ sets to generate by taking into account the range of the datatype properties. A reasonable approximation is that a datatype property from the source ontology that has for range the datatype U_i cannot be align to another datatype property from the target ontology that has for range the datatype U_j if $U_i \cap U_j = \emptyset$ (for instance *string* and *integer*). We remind that the possible datatypes considered by OWL are the hierarchy provided by XML Schema.

Finally, we operate a third approximation which turns out to be, unsurprisingly, the more efficient for reducing the space search: a local maximum heuristic. When forming a Σ set, we consider firstly a class from the first ontology, and gather all the entities (classes and properties) involved in its closure definition. We do the same for each classes of the second ontology and we evaluate all these small Σ sets for retaining the best one. We iterate this process over all the classes. Additional criteria allow us to guarantee the convergence of our approach (i.e. the order of the classes considered has no significance).

4.2 The Alignment API

In order to exchange and evaluate results of alignment algorithms, [8] has proposed a simple yet extensible alignment format. In first approximation, an alignment is a set of pairs of elements from each ontology. More deeply, a relation between entities of the source ontology and entities of the target ontology can be characterized. This relation is not restricted to the equivalence relation, but can be more sophisticated operators (e.g. subsumption, incompatibility, or even some fuzzy relation). A strength denotes the confidence held in this correspondence. Finally, an arity allows to note if the mapping is injective, surjective and total or partial on both side.

An API[5] has been developed for this format, with a default implementation, which eases the integration and the composition of new alignment algorithms, the generation of transformations and axioms, and the alignment comparison. All the classifiers detailed in the section 3 have been implemented to be compatible with

[5] `http://co4.inrialpes.fr/align/`.

this API, thus easing their chaining. Therefore, our *oMAP* framework benefits from all the evaluation facilities for comparing our approach with other methods as we will see in the next section.

5 Evaluation

The problem of aligning ontologies has already produced some interesting works. However, it is difficult to compare theoretically the various approaches proposed since they base on different techniques. Hence, it is necessary to compare them on common tests. This was the goal of an International Ontology Alignment Contest (EON) [24] who has set up benchmark tests for assessing the strengths and weakness of the available tools (section 5.1). We present the metrics used (section 5.2), and evaluate thoroughly our new approach with respect to the other competitors of this contest (section 5.3).

5.1 The EON Ontology Alignment Contest

The EON "Ontology Alignment Contest"[6] has been designed for providing some evaluation of ontology alignments algorithms. The evaluation methodology consisted in publishing a set of ontologies to be compared with another ontology. The participants were asked to provide the results in a particular format (see section 4.2). Along with the ontologies, a reference alignment was provided [24].

The set of tests consisted in one medium OWL ontology (33 named classes, 39 object properties, 20 data properties, 56 named individuals, and 20 anonymous individuals) to be compared to other ontologies. This initial ontology was about a very narrow domain (bibliographic references). There were three series of tests [24]:

- *simple tests*: compare the reference ontology with itself, with another irrelevant ontology or the same ontology in its restriction to OWL-Lite;
- *systematic tests*: obtained by discarding some features of the initial ontology leaving the remainder untouched. The considered features were (names, comments, hierarchy, instances, relations, restrictions, etc.).
- *complex tests*: four real-life ontologies of bibliographic references that were found on the web and left untouched.

5.2 Metrics

Standard information retrieval metrics are used to assess the different approaches. Let R the set of reference alignments ($|R|$ its cardinality), and A the set of alignments obtained by a certain method ($|A|$ its cardinality). The definitions of *precision* and *recall* are then given by: $Precision = |R \cap A|/|A|$ and $Recall = |R \cap A|/|R|$. Precision measures then the ratio between the number

[6] `http://oaei.inrialpes.fr/2004/Contest/`.

of correct alignments and the number of all mappings found, while recall measures the ratio between the number of correct alignments and the total number of correct mappings that should be found. Traditional precision and recall are defined in a analogous way, but with equality as similarity measure. In addition, we also combine precision and recall in the *F-measure* and the *overall-measure*: $F = 2 \cdot precision \cdot recall/(precision + recall)$ and $O = recall \cdot (2 - 1/precision)$.

5.3 Results and Discussion

There were four teams entering the EON initiative (Stanford/SMI, Fujitsu, INRIA & UMontréal and Karlsruhe) [24]. Among the three test sets, the most difficult one was the last one with real world (but above all various heterogeneity). The first one was quite easy. The second set of test was indeed able to help identifying where the algorithms were more handicapped (especially when they were unable to match strings).

Table 3. Precision and recall of oMAP with respect to the other competitors of EON

algo	**karlsruhe2**		**umontreal**		**fujitsu**		**stanford**		**oMAP**	
test	*Prec.*	*Rec.*	*Prec.*	*Rec.*	*Prec.*	*Rec.*	*Prec.*	*Rec.*	*Prec.*	*Rec.*
101	n/a	n/a	0.59	0.97	0.99	1.00	0.99	1.00	1.00	1.00
102	NaN	NaN	0.00	NaN	NaN	NaN	NaN	NaN	NaN	NaN
103	n/a	n/a	0.55	0.90	0.99	1.00	0.99	1.00	1.00	1.00
104	n/a	n/a	0.56	0.91	0.99	1.00	0.99	1.00	1.00	1.00
201	0.43	0.51	0.44	0.71	0.98	0.92	1.00	0.11	0.96	0.23
202	n/a	n/a	0.38	0.63	0.95	0.42	1.00	0.11	0.93	0.14
204	0.62	1.00	0.55	0.90	0.95	0.91	0.99	1.00	0.92	0.81
205	0.47	0.60	0.49	0.80	0.79	0.63	0.95	0.43	0.73	0.48
206	0.48	0.60	0.46	0.75	0.85	0.64	1.00	0.46	0.91	0.43
221	n/a	n/a	0.61	1.00	0.98	0.88	0.99	1.00	1.00	1.00
222	n/a	n/a	0.55	0.90	0.99	0.92	0.98	0.95	1.00	1.00
223	0.59	0.96	0.59	0.97	0.95	0.87	0.95	0.96	1.00	1.00
224	0.97	0.97	0.97	1.00	0.99	1.00	0.99	1.00	1.00	1.00
225	n/a	n/a	0.59	0.97	0.99	1.00	0.99	1.00	1.00	1.00
228	n/a	n/a	0.38	1.00	0.91	0.97	1.00	1.00	1.00	1.00
230	0.60	0.95	0.46	0.92	0.97	0.95	0.99	0.93	1.00	1.00
301	0.85	0.36	0.49	0.61	0.89	0.66	0.93	0.44	0.94	0.25
302	1.00	0.23	0.23	0.50	0.39	0.60	0.94	0.65	1.00	0.58
303	0.85	0.73	0.31	0.50	0.51	0.50	0.85	0.81	0.90	0.79
304	0.91	0.92	0.44	0.62	0.85	0.92	0.97	0.97	0.91	0.91

Table 3 gives the *precision/recall*, and Table 4, the *F-measure/O-measure* of *oMAP* with respect to the other competitors. Clearly, during the presentation of the results at the EON workshop, there were two groups of competitors and clear winners, since it seems that the results provided by Stanford and Fujitsu/Tokyo outperform those provided by Karlsruhe and Montréal/INRIA. We have developed our framework after this contest but use the same benchmark tests in order to compare our approach with the current best alignments. At first sight, we should be in the first group. In fact, it can be considered that these

constitute two groups of programs. The Stanford+Fujitsu programs are very different but strongly based on the labels attached to entities. For that reason they performed especially well when labels were preserved (i.e., most of the time). The Karlsruhe+INRIA systems tend to rely on many different features and thus to balance the influence of individual features, so they tend to reduce the fact that labels were preserved. Our mixed approach tend to success on both case even if we dispose yet of a large progression margin.

Table 4. F-Measure and overall-measure of oMAP with respect to the other competitors of EON

algo	**karlsruhe2**		**umontreal**		**fujitsu**		**stanford**		**oMAP**	
test	*F*	*O*	*F*	*O*	*F*	*O*	*F*	*O*	*F*	*O*
101	n/a	n/a	0.73	0.30	0.99	0.99	0.99	0.99	1.00	1.00
102	n/a	n/a	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
103	n/a	n/a	0.68	0.16	0.99	0.99	0.99	0.99	1.00	1.00
104	n/a	n/a	0.69	0.19	0.99	0.99	0.99	0.99	1.00	1.00
201	0.46	0.0-16	0.54	0.0-20	0.95	0.90	0.20	0.11	0.37	0.22
202	n/a	n/a	0.48	0.0-38	0.58	0.40	0.20	0.11	0.25	0.13
204	0.76	0.38	0.68	0.16	0.93	0.87	0.99	0.99	0.86	0.74
205	0.53	0.0-6	0.61	0.0-3	0.70	0.46	0.59	0.41	0.58	0.31
206	0.54	0.0-4	0.57	0.0-14	0.73	0.53	0.63	0.46	0.59	0.39
221	n/a	n/a	0.76	0.36	0.92	0.86	0.99	0.99	1.00	1.00
222	n/a	n/a	0.68	0.16	0.95	0.91	0.96	0.92	1.00	1.00
223	0.73	0.30	0.73	0.30	0.91	0.82	0.95	0.90	1.00	1.00
224	0.97	0.93	0.98	0.97	0.99	0.99	0.99	0.99	1.00	1.00
225	n/a	n/a	0.73	0.30	0.99	0.99	0.99	0.99	1.00	1.00
228	n/a	n/a	0.55	0.0-66	0.94	0.88	1.00	1.00	1.00	1.00
230	0.73	0.31	0.62	0.0-14	0.96	0.92	0.96	0.92	1.00	1.00
301	0.51	0.30	0.54	0.0-1	0.75	0.57	0.60	0.41	0.39	0.23
302	0.37	0.23	0.31	-1.0-20	0.47	0.0-35	0.77	0.60	0.74	0.58
303	0.79	0.60	0.38	0.0-60	0.51	0.02	0.83	0.67	0.84	0.71
304	0.92	0.83	0.51	0.0-17	0.89	0.76	0.97	0.95	0.91	0.82

6 Related Work

The alignment problem for ontologies, as well as the matching problem for schemas, has been addressed by many researchers so far and are strictly related. Some of the techniques applied in schema matching can be applied to ontology alignment as well, taking additionally into account the formal semantics carried out by the taxonomies of concepts and properties and the axioms of the ontology.

Related to schema matching are, for instance, the works [5, 6, 10, 16] (see [21] for a more extensive comparison). As pointed out above, closest to our approach is [10] based on a logical framework for data exchange, but we incorporated the classifier combinations (like GLUE) into our framework as well. GLUE [6, 5] employed a linear combination of the predictions of multiple base learners (classifiers). The combination weights are learned via regression on manually specified mappings between a small number of learning ontologies. The main improvement of our approach with respect to this system is then the structural

classifier which is able to align two ontologies solely on their semantics, and without the presence of individuals.

Among the works related to ontology alignment, [7] propose to combine different similarity measures from pre-existing hand-established mapping rules. Besides the validity of these rules could be generally put into question, this method suffers from not being fully automatic. [18] has developed an interesting approach: from anchor-pairs of concepts that seem to be close (discovered automatically or proposed manually), their *hors-context* similarity are computed analyzing the paths in the taxonomy that link the pairs of concepts. This method has been implemented into the ANCHOR-PROMPT tool which has, until now, one of the best performance (see section 5.3). [9] have adapted works on similarity calculus for object-based knowledge representation languages to the Semantic Web languages. A global similarity measure taking into account all the features of the OWL-Lite language has been proposed, capable to treat both the circular definitions and the collections. For a complete state of the art on the numerous ontology alignment approaches proposed, see [15].

7 Conclusion and Future Work

As the number of Semantic Web applications is growing rapidly, many individual ontologies are created. The development of automated tools for ontology alignment will be of crucial importance. In this paper, we have presented a formal framework for ontology Matching, which for ease we call *oMap*. oMap uses different classifiers to estimate the quality of a mapping. Novel is the classifier which uses the structure of the OWL constructs and thus the semantics of the entities defined in the ontologies. We have implemented the whole framework and evaluated it on independent benchmark tests provided by the EON "Ontology Alignment Contest" [24] with respect to the other competitors.

As future work, we see some appealing points. The combination of a rule-based language with an expressive ontology language like OWL has attracted the attention of many researchers [13] and is considered now as an important requirement. Taking into account this additional semantics of the ontologies appear thus necessary. Additional classifiers using more terminological resources can be included in the framework, while the effectiveness of the machine learning part could be improved using other measures like the KL-distance. While to fit new classifiers into our model is straightforward theoretically, practically finding out the most appropriate one or a combination of them is quite more difficult. In the future, more variants should be developed and evaluated to improve the overall quality of *oMAP*. Furthermore, instead of averaging the classifier predictions, the appropriateness of each classifier could be learned via regression. These ideas are currently investigated since we are participating actively to the 2005 campaign of the Ontology Alignment Evaluation Initiative[7].

[7] `http://oaei.inrialpes.fr/`

References

1. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P.F.: The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press, 2003.
2. Bachimont, B., Isaac, A., Troncy, R.: Semantic Commitment for Designing Ontologies: A Proposal. In *13th Int. Conf. on Knowledge Engineering and Knowledge Management (EKAW'02)*, Sigüenza, Spain, 2002, 114–121.
3. Bechhofer, S., Volz, R., Lord, P.W.: Cooking the Semantic Web with the OWL API. In *2nd Int. Semantic Web Conf. (ISWC'03)*, Sanibel Island, Florida, USA, 2003, 659–675.
4. Berners-Lee, T., Hendler, J., Lassila, O.: The Semantic Web. Scientific American, **284**(5), 2001, 34–43.
5. Dhamankar, R., Lee, Y., Doan, A., Halevy, A., Domingos, P.: iMap: discovering complex semantic matches between database schemas. In *ACM SIGMOD Int. Conf. on Management of Data*, Paris, France, 2004, 383–394.
6. Doan, A., Madhavan, J., Dhamankar, R., Domingos, P., Halevy, A.: Learning to Match Ontologies on the Semantic Web. The VLDB Journal, **12**(4), 2003, 303–319.
7. Ehrig, M., Staab, S.: QOM - quick ontology mapping. In *3rd Int. Semantic Web Conf. (ISWC'04)*, Hiroshima, Japan, 2004, 683–697.
8. Euzenat, J.: An API for ontology alignment. In *3rd Int. Semantic Web Conf. (ISWC'04)*, Hiroshima, Japan, 2004, 698–712.
9. Euzenat, J., Valtchev, P.: Similarity-based ontology alignment in OWL-Lite. In *15th European Conf. on Artificial Intelligence (ECAI'04)*, Valence, Spain, 2004, 333–337.
10. Fagin, R., Kolaitis, P.G., Miler, R.J., Popa, L.: Data Exchange: Semantics and Query Answering. In *9th Int. Conf. on Database Theory (ICDT'03)*, Sienna, Italy, 2003, 207–224.
11. Hahn, U., Cornet, R., Schulz, S. editors. *KR-MED: 1st Int. Workshop on Formal Biomedical Knowledge Representation*. Whistler, Canada, 2004.
12. Horrocks, I., Patel-Schneider, P.F., van Harmelen, F.: From SHIQ and RDF to OWL: The making of a web ontology language. Journal of Web Semantics, **1**(1), 2003, 7–26.
13. Horrocks, I., Patel-Schneider, P.F.: A proposal for an OWL rules language. In *13th Int. World Wide Web Conf. (WWW'04)*, New York, USA, 2004, 723–731.
14. Isaac, A., Troncy, R.: Designing and Using an Audio-Visual Description Core Ontology. In *Workshop on Core Ontologies in Ontology Engineering at EKAW'04*, Whittlebury Hall, Northamptonshire, UK, 2004.
15. KW Consortium: State of the Art on Ontology Alignment. Deliverable Knowledge Web 2.2.3, FP6-507482, 2004.
16. Madhavan, J., Bernstein, P.A., Rahm, E.: Generic schema matching with cupid. In *27th Int. Conf. on Very Large Data Bases (VLDB'01)*, Roma, Italy, 2001, 49–58.
17. Nottelmann, H., Straccia, U.: sPLMap: A probabilistic approach to schema matching. In *27th European Conf. on Information Retrieval (ECIR'05)*, Santiago de Compostela, Spain, 2005, 81–95.
18. Noy, N.F., Musen, M.A.: Anchor-PROMPT: Using non-local context for semantic matching. In *Workshop on Ontologies and Information Sharing at IJCAI'01*, Seattle, Washington, USA, 2001.
19. OWL, Web Ontology Language Reference Version 1.0. W3C Recommendation, 10 February 2004. `http://www.w3.org/TR/owl-ref/`

20. Porter, M.F.: An algorithm for suffix stripping. Program, **14**(3), 1980, 130–137.
21. Rahm, E., Bernstein., P.A.: A survey of approaches to automatic schema matching. The VLDB Journal, **10**(4), 2001, 334–350.
22. RDF, Ressource Description Framework Primer W3C Recommendation, 10 February 2004. `http://www.w3.org/TR/rdf-primer/`
23. Sebastiani, F.: Machine learning in automated text categorization. ACM Comuting Surveys, **34**(1), 2002, 1–47.
24. Sure, Y., Corcho, O., Euzenat, J., Hughes, T. editors. *3rd Int. Workshop on Evaluation of Ontology-based Tools (EON'04)*, Hiroshima, Japan, 2004.