Ontology alignment with OLA

Jérôme Euzenat [1], **David Loup** [2], **Mohamed Touzani** [3], **Petko Valtchev** [4]

Abstract

Using ontologies is the standard way to achieve interoperability of heterogeneous systems within the Semantic web. However, as the ontologies underlying two systems are not necessarily compatible, they may in turn need to be aligned. Similarity-based approaches to alignment seems to be both powerful and flexible enough to match the expressive power of languages like OWL. We present an alignment tool that follows the similarity-based paradigm, called OLA. OLA relies on a universal measure for comparing the entities of two ontologies that combines in a homogeneous way the entire amount of knowledge used in entity descriptions. The measure is computed by an iterative fixed-point-bound process producing subsequent approximations of the target solution. The alignments produce by OLA on the contest ontology pairs and the way they relate to the expected alignments is discussed and some preliminary conclusions about the relevance of the similarity-based approach as well as about the experimental settings of the contest are drawn.

1 Presentation of the system

The ontology alignment tool OLA (for *O*WL-*L*ite *A*lignment) is jointly developed by the teams at DIRO, University of Montréal and INRIA Rhône Alpes. Its main features are presented below (see also [5, 6]).

1.1 General purpose statement

OLA is dedicated to the alignment of ontologies expressed in OWL, with an emphasis on its restricted dialect, called OWL-Lite.

1.1.1 Functional specifications

More than a simple tool for automated alignment construction, OLA is designed as an environment for manipulating alignments. Indeed, in its current version, the system offers the following services:

- parsing and visualization of (pairs of) ontologies,
- automated computation of similarities between entities from different ontologies,
- automated extraction of alignments from a pair of ontologies,
- manual construction of alignments,
- initialization of automated alignment construction by an existing alignment,
- visualization of alignments,
- comparison of alignments.

In the following sections we focus exclusively on the automated alignment construction and the related services offered by OLA.

[1] INRIA Rhône-Alpes, `Jerome.Euzenat@inrialpes.fr`
[2] Université de Montréal
[3] Université de Montréal
[4] Université de Montréal, `Petko.Valtchev@umontreal.ca`

1.1.2 Basic assumptions

Universality : All the available knowledge about ontology entities should be taken into account when aligning.

Automation : We required the alignment mechanisms to be of the highest possible automation degree. In other terms, although the entire alignment process may be set on a semi-automated basis, the production of an alignment should not require user intervention at the intermediate steps. Thus, we expect the user to provide a minimal set of parameters for the alignment process whereas the tool will suggest one or more candidate alignments at the end. This may be performed in a loop aimed at establishing the optimal parameters for a specific case or domain. Automation may be used for optimal parameter learning as well.

Uniform comparison : Following the syntactic structure of the OWL language, entities are divided into categories, e.g., *classes*, *objects*, *properties*, *relations*, and only entities of the same category are compared. Moreover, the entities of a category are compared using the same similarity function and on the same feature space. In other words, for each pair of entities of a given category, the same set of similarity factors are considered and the respective contributions of those factors to the overall similarity of the pair are combined in a way that depends only on the category.

Comparability of similarity results : To enable comparison of similarity scores between different alignment tasks, the values of the similarity measure are normalized. It is noteworthy that normalization is enforced throughout the entire iterative computation process via an appropriate function definition. Moreover, useful properties of the function as proximity measure are ensured such as *positiveness*, maximalness[5], and *symmetry*[6].

1.1.3 Specific restrictions

- Primary focus is on a rather restricted sublanguage, OWL-Lite. However, some constructs from the richer OWL-DL are also supported. As a long-term goal, the coverage of the entire OWL-DL language will be sought.
- No inference is performed on the ontology, in particular inheritance is not used to expand entity descriptions. This choice has been motivated by efficiency considerations. It could be easily altered by applying a limited form of reasoning to the available descriptive knowledge as a pre-processing step.
- Only descriptive knowledge is taken into account: The similarity of an entity pair depends on all the similarities of neighbor pairs whose members **describe** the respective initial entities. In other terms, given two neighbor entities e_1 and e_2, e_2 may appear in a similarity expression for e_1 if the link between both is considered as a part of the description of e_1. For instance, we consider that a data type is not described by a property whose range the datatype represents. Consequently, datatypes are compared in an ontology-independent manner.
- Entity category separation is enforced in similarity definition: Only entities from the same category are compared for similarity and hence for alignment. Thus, classes from the first ontology are compared to classes from the second one, and datatypes to datatypes, respectively[7].

1.2 Specific techniques used

OLA relies on an all-encompassing similarity measure that is defined by a system of quasi-linear equations. Its actual values are computed as the fixed point of an iterative approximation process which starts with a lexical similarity measure and gradually brings in contributions from structure comparing functions. The entire computation process is supported by a graphical representation of the ontology structure, the OL-Graph of the ontology.

1.2.1 OL-Graph construction

To provide an easy-to-process inner representation of OWL ontologies, we use graph structure that we called OL-Graph. An OL-Graph is a labeled graph where vertices correspond to OWL

[5] With normalization, this amounts to forcing scores of 1 for identical entities within identical ontologies

[6] The price to pay for symmetry is the impossibility of detecting subsumption by this purely numerical procedure.

[7] However, some test cases, e.g., the alignment of ontology 301, suggest that a class may be more advantageously aligned to a datatype.

entities and edges to inter-entity relationships. As described in [6], the set of different vertex categories is: class (C), object (O), relation (R), property (P), property instance (A), datatype (D), datavalue (V), property restriction labels (L). Moreover, distinction is made between datatype relations (R_{dt}) and object relations (R_o), as well as between datatype properties (P_{dt}) and object ones (P_o).

The relationships expressed in the OL-Graph are:

- *specialization* between classes or relations ($\mathcal{S}$),
- *instanciation* ($\mathcal{I}$) between objects and classes, property instances and properties, values and datatypes,
- *attribution* ($\mathcal{A}$) between classes and properties, objects and property instances;
- *restriction* ($\mathcal{R}$) expressing the restriction on a property in a class,
- *valuation* ($\mathcal{U}$) of a property in an object.

The OL-Graph of an ontology is constructed after the ontology is parsed and its entities and their relationships extracted. So far, we use the OWL API [1] for the parsing of OWL files, but other possibilities remain open. It is noteworthy that OL-Graph supports well inference process. For instance, the graphs of an ontology can be easily extended with the descriptive knowledge derived by inheritance between classes or relations, or by saturation following the property types (e.g., by adding transitivity arcs for a `owl:TransitiveProperty`). Further details on OL-Graph construction will be given in [8]. It does not, however, scale to OWL-Full.

1.2.2 Integrative similarity measure

Our similarity model assigns a specific function to each node category in the OL-Graph. The functions are designed in a way to cover the greatest possible part of the available descriptive knowledge about a couple of entities. Thus, given a category X, the similarity of two nodes from X depends on:

- the similarities of the terms used to designate them (may be URIs, labels, names, etc.),
- the similarity of the pairs of neighbor nodes in the respective OL-Graphs that are linked by edges expressing the same relationships (e.g., class node similarity depends on similarity of superclasses, of property restrictions and of member objects),
- the similarity of other local descriptive features depending on the specific category (e.g., cardinality intervals, property types)

Datatype and datavalue similarities are external to our model. As such, they are provided by the user or measured by a standard function (e.g., string identity of values and datatype names/URIs).

Formally, given a category X together with the set of relationships it is involved in, $\mathcal{N}(X)$, the similarity measure $Sim_X : X^2 \rightarrow [0, 1]$ is defined as follows:

$$Sim_X(x, x') = \sum_{\mathcal{F} \in \mathcal{N}(X)} \pi^X_{\mathcal{F}} MSim_Y(\mathcal{F}(x), \mathcal{F}(x')).$$

The function is normalized, i.e., the weights $\pi^X_{\mathcal{F}}$ sum to a unit, $\sum_{\mathcal{F} \in \mathcal{N}(X)} \pi^X_{\mathcal{F}} = 1$. for the computability For instance, for two classes c, c' :

$$\begin{aligned} Sim_C(c, c') = {} & \pi^C_L sim_L(\lambda(c), \lambda(c')) \\ & + \pi^C_{\mathcal{I}} MSim_O(\mathcal{I}(c), \mathcal{I}'(c')) \\ & + \pi^C_{\mathcal{S}} MSim_C(\mathcal{S}(c), \mathcal{S}'(c')) \\ & + \pi^C_{\mathcal{A}_{dt}} MSim_P(\mathcal{A}_{dt}(c), \mathcal{A}'_{dt}(c')) \\ & + \pi^C_{\mathcal{A}_o} MSim_P(\mathcal{A}_o(c), \mathcal{A}'_o(c')) \end{aligned}$$

The set functions $MSim_Y$ compare two sets of nodes of the same category. They are presented in the next paragraph. Table 1 illustrates the set of similarities in our model.

1.2.3 Similarity-based matching of entity sets

In order to ensure equity between factors in $Sim_X(n_1, n_2)$, all similarities of pairs linked with the same type of links are combined into a unique value. This is achieved by means of a generic set similarity function $MSim$. Its arguments are two sets S_1 and S_2 of entities of the same category Y and the respective measure Sim_Y. The result is an average of the similarities of a

Funct.	Node	Factor	Measure
Sim_O	$o \in O$	$\lambda(o)$	sim_L
		$a \in A, (o, a) \in \mathcal{A}$	$MSim_A$
Sim_A	$a \in A$	$r \in R, (a, r) \in \mathcal{R}$	Sim_R
		$o \in O, (a, o) \in \mathcal{U}$	$MSim_O$
		$v \in V, (a, v) \in \mathcal{U}$	$MSim_V$
Sim_V	$v \in V$	value literal	type dependent
Sim_C	$c \in C$	$\lambda(c)$	sim_L
		$p \in P, (c, p) \in \mathcal{A}$	$MSim_P$
		$c' \in C, (c, c') \in \mathcal{S}$	$MSim_C$
sim_D	$d \in D$	$\lambda(r)$	XML-Schema
Sim_R	$r \in R$	$\lambda(r)$	sim_L
		$c \in C, (r, \texttt{domain}, c) \in \mathcal{R}$	$MSim_C$
		$c \in C, (r, \texttt{range}, c) \in \mathcal{R}$	$MSim_C$
		$d \in D, (r, \texttt{range}, d) \in \mathcal{R}$	Sim_D
		$r' \in R, (r, r') \in \mathcal{S}$	$MSim_R$
Sim_P	$p \in P$	$r \in R, (p, r') \in \mathcal{S}$	Sim_R
		$c \in C, (p, \texttt{all}, c) \in \mathcal{R}$	$MSim_C$
		$n \in \{0, 1, \infty\}, (p, \texttt{card}, n) \in \mathcal{R}$	equality

Table 1. Similarity function decomposition (`card` = `cardinality` and `all` = `allValuesFrom`).

limited subset of the product $S_1 \times S_2$. The subset represents a matching that optimizes the total similarity [9]:

$$MSim_C(S_1, S_2) = \frac{\sum_{\langle c_1, c_2 \rangle \in Pairing(S_1, S_2)} Sim_C(c_1, c_2)}{max(|S_1|, |S_2|)}$$

where $Pairing(S_1, S_2)$ is the optimal matching. For normalization reasons, the sum of the similarities of the pairs in $Pairing(S_1, S_2)$ is divided by the size of the larger set.

1.2.4 Equation system definition and iterative resolution

As many of the relationships are both-ways, it may be impossible to follow standard procedures in computing the similarity values. Indeed, the recursive definition of the similarity may easily lead to circular dependancies of the similarity values for two or more node pairs. In such cases, an equation system is composed (see [2, 9]) out of the similarity definitions where variables correspond to similarities of node pairs while coefficients come from weights.

Because of the uncertainty due to the matching functions whose outcome cannot be fixed *a priori*, the resulting system is not linear and therefore cannot be solved in a direct way. Instead, an iterative method is used to approximate the solution (which always exist) as the fixed point of a vector function. The process starts with the local similarity, i.e., the one computed without looking at neighbor nodes. It then integrates neighbor similarities and lets them grow as a result of mutual influence. The growth is steady at each step of the iterative process but is nevertheless limited from above since neither of the functions from table 1 can reach values greater than 1. Thus, the process necessarily ends with a vector fixed point whose components are the similarity values sought.

1.2.5 Lexical similarity measures

OLA relies on WordNet 2.0 [7] for comparing identifiers. For that purpose, it applies a measure of "relatedness" between two terms. Given a pair of identifiers, the lexical similarity mechanisms retrieve the sets of synonyms (the *synsets*) for each term. A normalized Hamming distance is then applied to these sets. A variant of the substring distance is used to establish a default similarity value for identifier pairs. Such a default mechanism allows identifiers that are not entries in WordNet, e.g., compound identifiers or abbreviations, to be processed in a sensible way.

1.3 Implementation

OLA is implemented in JAVA. Its architecture follows the one of the Alignment API and the recent implementation that was described in [4]. OLA relies on the OWL API [1] for parsing OWL files. An entire subsystem is dedicated to the onstruction of OL-Graphs on top of the parsed ontologies. A set of further components offer similarity computation services: substring distances, edit distances, Hamming distance, etc. A specific component extracts similarity values from the limited WordNet interface provided by the JWNL library [3]. Similarity-based matching between sets of entities is performed by another component. Similarity and matching mechanisms are integrated into the alignment producing subsystem which supports the entire iterative computation

process. Finally, the VISON subsystem provides a uniform interface to all the automated tools and visualizes both the input data, i.e., the OL-Graphs, and the final result, i.e., the alignment.

1.4 Adaptation made for the proposal

Several changes have been made to fit the complexity of the comparison. The most noteworthy one is the abandon of the requirement that all entities of the same category are compared along the same feature space.

1.4.1 Adaptive description space

We found that the "uniform factor weights" condition tends to favor pairs of entities that have complete descriptions, i.e., pairs where both the members are connected to at least one descriptive entity for each of the similarity factors in the respective formula. Conversely, pairs where a particular factor is void tend to score to lesser similarity values. The extreme case is the pair of `Thing` classes which, if present, usually have almost no description. With fixed weights for similarity factors, and hence universal feature space for comparison, the `Thing` class pair will be evaluated to a relatively weak similarity value and the chances are high for it to be skipped from the alignment.

For the above reasons, we decided to limit the comparison of two entities to the strict set of factors which are non void in both. This has been achieved in an uniform way, i.e., through a division of the initial linear combination formula by the sum of the weights of all non-void factors. Thus, for a category X, the similarity measure $Sim_X^+ : X^2 \rightarrow [0, 1]$ becomes:

$$Sim_X^+(x, x') = \frac{Sim_X(x, x')}{\sum_{\mathcal{F} \in \mathcal{N}^+(x,x')} \pi_{\mathcal{F}}}$$

where $\mathcal{N}^+(x, x')$ is the set of all relationships $\mathcal{F}$ for which $\mathcal{F}(x) \cup \mathcal{F}(x') \neq \emptyset$ [8].

1.4.2 Lexical similarity measure

The initial straightforward similarity measure has been replaced by a more sophisticated one that better accounts for semantic proximity between compound identifiers. Thus, given a pair of identifiers, they are first "tokenized", i.e., split into a set of atomic terms. Then, the respective pairs of terms are compared using WordNet. In fact, their degree of relatedness is computed as the ratio between the depth of the most specific common hypernym and the sum of both term depths. Finally, a similarity-based match is performed to establish a degree of proximity between the sets of terms.

1.4.3 Weight finding mechanism

To increase the level of automation in OLA, a weight-search mechanism was added to the initial architecture. Indeed, it is far from obvious for a novice user how to weight the different similarity factors. The underlying module performs several runs of the alignment producing subsystem with various weight combinations. It keeps only the combination that has resulted in the best alignment, i.e., the one of the highest total similarity between aligned entities. On the one hand, this procedure is not realistic in a setting where reference alignments are not given. On the other hand, if the tests a realistic, then what is learned is the best behaviour of the system in general.

2 Results

The test protocol was as follows. We first looked for the typical weight combinations with an exhaustive search on a small subset of test cases. The resulting combinations were then applied systematically to the rest of the ontology pairs. Whenever the results were unsatisfactory, exhaustive search was applied to the neighborhood of the best scoring typical combinations. Here we provide some details on the combinations that were mined out by OLA as well as a brief comment for every single test indicating the combination of parameters that led to the best scoring alignment. A summary of the results obtained with equal weights for all factors in a category is provided at the end as well.

[8] That is, exists at least one y such that $(x, y) \in \mathcal{F}$ or at least one y' such that $(x', y') \in \mathcal{F}$.

2.1 Preliminary tests

The optimal weight searching engine of OLA was run on a small subset of ontologies that seemed to represent the extreme cases. The resulting matchings were compared to the respective expected alignments according to the contest guidelines. The underlying weight combinations and their respective alignment scores were then analyzed to discover possible trends. For this preliminary experiment, the step in the variation of the specific weight values was set to 0.2 while the total of all weights in a category was set to 1. This value provided a good trade-off between the range of variation for each single weight (i.e., a five-grade scale) and the number of combinations to be tested. Actually, there are 8 categories with 3, 4 or 5 weights. To bring down the resulting combinatorial explosion, we used the same weight combination for entity categories sharing the same set of similarity factors, e.g., datatype and object properties.

The results of this step suggested that there were three weight combinations that can lead to the best scoring alignment for a test case:

- equal or nearly equal weights for all factors,
- one factor is assigned the total weight of 1 while the other weights are set to 0,
- the total weight of 1 is divided into two non-zero parts assigned to two factors, the remaining factors are given zero weights.

In what follows, we indicate for each test the weight combination that led to the best alignment with respect to precision and the lexical similarity used. To provide an idea about the average performances of OLA, we include also a summary of the scores obtained with perfectly equal weights in every entity category (i.e., 0.2-step constraint relaxed). It is noteworthy that the overwhelming majority of the results where precision is below 1.0 are mere lower bounds and may well be improved through an exhaustive search in the weight combination space. Moreover, in each test, our tool aligned all the named entities of the basic ontology to the most similar entity of the compared ontology. Therefore, the recall scores, which depend on the size of the proposed alignment, are relatively low.

2.2 Concept

In this group of tests, the string distance was systematically used for lexical comparisons.

2.2.101 Identity

The best alignment was obtained with unit weight to lexical similarity and zero weights to the remaining factors for all categories.

Precision	Recall	Fallout
1.0	0.611	0.0

2.2.102 Irrelevant ontology

OLA used equal weights to obtain the following result that proved best.

Precision	Recall	Fallout
1.0	N/A	0.0

2.2.103 Language generalization

The best combination assigns for each category 0.4 weight to the lexical similarity and 0.6 to the factor representing the links to more general entities (e.g., the super classes of a class, the class for an individual, the relation for a property restriction, etc.).

Precision	Recall	Fallout
1.0	0.611	0.0

2.2.104 Language restriction

The combination that scored best is identical to the one described in the previous paragraph.

Precision	Recall	Fallout
1.0	0.611	0.0

2.3 Systematic

2.3.201 No names

Equal weights were used together with string distance.

Precision	Recall	Fallout
0.714	0.436	0.286

2.3.202 No names, no comment

The same settings as in test 201 were used.

Precision	Recall	Fallout
0.626	0.383	0.374

2.3.204 Naming conventions

are used for labels.

The same settings as in test 201 were used.

Precision	Recall	Fallout
0.901	0.550	0.099

2.3.205 Synonyms

Equal weights and WordNet led to the best precision alignment.

Precision	Recall	Fallout
0.802	0.490	0.198

2.3.206 Foreign names

The settings used were identical to those of test 205.

Precision	Recall	Fallout
0.761	0.450	0.239

2.3.221 No hierarchy

The settings used were identical to those of test 205.

Precision	Recall	Fallout
1.0	0.611	0.0

2.3.222 Flattened hierarchy

The best combination is equal to the one for test 201, except for the class category where the 0.6 weight was assigned to the instance factor. String distance was used as well.

Precision	Recall	Fallout
0.945	0.577	0.055

2.3.223 Expanded hierarchy

The same settings as in test 222 were used, with the exception of the 0.6 weight in the class category which was assigned to the datatype property factor.

Precision	Recall	Fallout
0.989	0.604	0.011

2.3.224 No instances

The same settings as in test 205.

Precision	Recall	Fallout
1.0	0.968	0.0

2.3.225 No restrictions

The same settings as in the test 222 were used.

Precision	Recall	Fallout
1.0	0.611	0.0

2.3.228 No properties

Once again, the winning combination had equal weights for all factors with string distance.

Precision	Recall	Fallout
1.0	0.375	0.0

2.3.230 Flattening entities

The winning weight combination was the one of test 222 but with WordNet-based similarity.

Precision	Recall	Fallout
0.946	0.476	0.054

2.4 Real ontologies

2.4.301 BibTeX/MIT

The exclusively lexical comparison, e.g., weight of 1.0 to the lexical similarity factors in all categories, which was supported by WordNet produced the best alignment in this case.

Precision	Recall	Fallout
0.623	0.513	0.377

2.4.302 BibTeX/UMBC

Same settings as in the 301 test.

Precision	Recall	Fallout
0.542	0.245	0.458

2.4.303 Karlsruhe

The same settings as in test 205.

Precision	Recall	Fallout
0.5	0.311	0.5

2.4.304 INRIA

Same settings as in the 301 test.

Precision	Recall	Fallout
0.671	0.315	0.329

2.5 Summary of equal-weight results

Figure 1 summarizes the results obtained by OLA with equal weight combinations.

Test Nbr	Name	Lex. Sim.	Precision	Recall	Fallout
101	Id	SD	0.97	0.59	0.03
102	Irrelevant	SD	1.0	N/A	0.0
103	Language Generalisation	SD	0.901	0.550	0.099
104	Language Restriction	SD	0.912	0.557	0.088
201	No Names	SD	0.714	0.436	0.286
202	No Names, No Comments	SD	0.626	0.383	0.374
204	Naming Conventions	SD	0.901	0.550	0.099
205	Synonyms	WN	0.802	0.490	0.198
206	Foreign Names	WN	0.761	0.450	0.239
221	No Hierarchy	WN	1.0	0.611	0.0
222	Flattened Hierarchy	WN	0.901	0.550	0.099
223	Expanded Hierarchy	WN	0.967	0.590	0.033
224	No Instances	WN	1.0	0.968	0.0
225	No Restrictions	WN	0.967	0.590	0.033
228	No Properties	SD	1.0	0.375	0.0
230	Flattening Entities	WN	0.92	0.463	0.08
301	BibTeX/MIT	WN	0.607	0.493	0.393
302	BibTeX/UMBC	WN	0.5	0.226	0.5
303	Karlsruhe	WN	0.5	0.311	0.5
304	INRIA	WN	0.618	0.439	0.382

Figure 1. Results of the alignment with equal weights. Lexical similarity codes: WN stands for similarity based on WordNet and SD for (inverted) string distance.

3 General comments

3.1 Comments on the results (strength and weaknesses)

According to experimental results, our algorithm performs well when the structure of the compared ontologies are closed or identical (tests 10X and 22X).

3.2 Discussions on the way to improve the proposed system

Many of the initial assumption and constraints have proven to be a hamper for the establishment of precise alignments. Here is a discussion of points that could not be corrected during the test period but that we shall look at in the aftermath.

3.2.1 Limited inference in OL-Graph construction

In the construction process of our inner representation of the ontology we plan to expand class, relation and property nodes with the description knowledge they inherit from the super entities. Similarly, in the case where class restrictions fix property values, these values will be brought down to the descriptions of class instances.

3.2.2 Inter-category comparisons

An extended version of the similarity measure should allow the comparison of entities from different categories:

- classes with data types,
- object properties with datatype ones,
- objects with values.

3.3 Comments on the test cases

We fond that the proposed testbed cases cover a large portion of the situations that may arise in ontology alignment practice. However, the targeted variation of the test cases, i.e., on one specific dimension at once, is a challenge for our algorithm. Indeed, it was designed to be robust

on all features, hence no single feature is favored by the collection of weights. This does not seem to be a winning strategy with a test set that alters systematically a single feature: Whereas an algorithm that puts the emphasis on a particular feature will be negatively affected only by the test that puts noise on that feature, our own system experiences a systematic, albeit much lighter, negative impact.

3.4 Comments on measures

The proposed measures are definitely a first step in the right direction. Applying information retrieval metrics such as recall and precision seems to be a good approximation for the expectations of an alignment tool user. And the underlying model is simple enough to be understood by an average user. However, there is a price to pay for the simplicity, in particular with similarity-based alignment tool that grades the strength of an alignment cell. In fact, the counting of "hits" and "misses" ignores completely the actual strength values which may vary in large ranges.

3.5 Proposed new measures

A possibility would be to integrate the strength of the cells in the precision computation, something that could be done at low cost (e.g., by adding up strength values instead of counting).

4 Raw results

4.1 Links to the set of provided alignments (in align format)

A .zip archive of all the mentioned results together with indication of the OLA settings used for their extraction is provided at:
`http://www.iro.umontreal.ca/~owlola/align_files.html`.

4.2 Matrix format

See section 2.5.

5 Conclusions

It is still too early to draw final conclusions on the capacity of our system and our similarity-based approach in general to produce meaningful alignments on real ontologies. Indeed, the results on artificially altered ontologies only suggest the tool is robust to a single, albeit often powerful, source of noise. Further experiments will be necessary to gain deeper insight into the behavior of our alignment mechanisms. It is also noteworthy that the time allocated to the preparation of the contest was clearly insufficient to deal with all the challenging issues it has revealed. We are nevertheless very obliged to the contest organizers for their excellent initiative. Indeed, our participation effort yielded a long list of exciting problems to look at.

Despite the partiality of the picture we could draw about the performances of OLA, we would advocate for similarity as a mechanism for supporting alignment construction. It represents a good trade-off between several criteria that need to be taken into account in the design of effective alignment tools: precision of the final results, computational efficiency, good level of automation.

REFERENCES

[1] Sean Bechhofer, Raphael Voltz, and Phillip Lord. Cooking the semantic web with the OWL API. In *Proc. 2nd International Semantic Web Conference (ISWC), Sanibel Island (FL US)*, 2003.

[2] Gilles Bisson. Learning in FOL with similarity measure. In *Proc. 10th AAAI, San-Jose (CA US)*, pages 82–87, 1992.

[3] John Didion. The Java WordNet Library, http://jwordnet.sourceforge.net/, 2004.

[4] Jérôme Euzenat. An API for ontology alignment. In *Proc. 3rd ISWC*, pages 698–712, Hiroshima (JP), 2004.

[5] Jérôme Euzenat and Petko Valtchev. An integrative proximity measure for ontology alignment. In *Proc. ISWC-2003 workshop on semantic information integration, Sanibel Island (FL US)*, pages 33–38, 2003.

[6] Jérôme Euzenat and Petko Valtchev. Similarity-based ontology alignment in OWL-lite. In *Proc. 15th ECAI*, pages 333–337, Valencia (ES), 2004.

[7] A.G. Miller. Wordnet: A lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

[8] Mohamed Touzani. Alignement d'ontologies dans OWL. Master's thesis, University of Montréal, in preparation.

[9] Petko Valtchev. *Construction automatique de taxonomies pour l'aide à la représentation de connaissances par objets*. Thèse d'informatique, Université Grenoble 1, 1999.