Ontology matching benchmarks: generation and evaluation

Maria Roşoiu, Cássia Trojahn, and Jérôme Euzenat

INRIA & LIG, Grenoble, France
Firstname.Lastname@inria.fr

Abstract. The OAEI Benchmark data set has been used as a main reference to evaluate and compare matching systems. It requires matching an ontology with systematically modified versions of itself. However, it has two main drawbacks: it has not varied since 2004 and it has become a relatively easy task for matchers. In this paper, we present the design of a modular test generator that overcomes these drawbacks. Using this generator, we have reproduced Benchmark both with the original seed ontology and with other ontologies. Evaluating different matchers on these generated tests, we have observed that (a) the difficulties encountered by a matcher at a test are preserved across the seed ontology, (b) contrary to our expectations, we found no systematic positive bias towards the original data set which has been available for developers to test their systems, and (c) the generated data sets have consistent results across matchers and across seed ontologies. However, the discriminant power of the generated tests is still too low and more tests would be necessary to draw definitive conclusions.
Keywords: Ontology matching, Matching evaluation, Test generation, Semantic web.

1 Introduction

Evaluating ontology matching may be achieved in several ways. The most common one consists of providing matchers with two ontologies and comparing the returned alignment with a reference alignment [4]. However, this raises the issue of the choice of ontologies and the validity of the reference.

Since 2004, the Ontology Alignment Evaluation Initiative (OAEI)[1] makes available a collection of data sets for evaluating matching systems. One such data set is Benchmark. It is a well-defined set of tests in which each test is composed of two ontologies and a reference alignment. The tests are based on one particular ontology, from the bibliographic domain, and systematic alterations of this ontology, e.g., removing classes, renaming properties.

Benchmark was designed with the aim of covering the problem space, i.e., the various situations in which a matcher may be. However, this data set has various drawbacks: (a) lack of realism: tests are mechanically generated and cover

[1] http://oaei.ontologymatching.org/

a systematic alteration space, (b) lack of variability: it always uses the same seed ontology altered in the exact same way, and (c) lack of discriminability: the tests are not difficult enough to discriminate well matchers.

We are not particularly interested in Drawback (a) because it has been overcame by other data tests made available by OAEI. We focus on drawbacks (b) and (c). To that extent, we have developed a test generator that may be used with any seed ontology and allows for fine tuning the input parameters, as well as randomized modifications over the ontology entities. A byproduct of this generator is that it enables us to evaluate the relevance of the Benchmark dataset: by reproducing this dataset and using it to evaluate different matchers in the same conditions, we can assess how much the results obtained are dependent on the particular seed ontology or the particular matcher.

We run different matchers on the generated tests, which allows us to draw conclusions on the results obtained so far with Benchmark:

- The difficulties encountered by a matcher at a test are preserved across the seed ontology, hence, Benchmark is relevant.
- Matchers have, in general, no better results with the original Benchmark than with the new generated data sets, this goes counter our expectation that, because tests and results were available, matchers would perform better.
- Matcher results are generally consistent across seed ontologies and ontology results are generally consistent across matchers, but with low discrimination. This confirm that the lack of discriminability is due to Benchmark and not to the seed ontology.

The rest of the paper is structured as follows. In Section 2, we present the state-of-the-art in ontology matching test generation. In Section 3, we present the architecture of our test generator and the strategy we came with in order to reproduce the Benchmark dataset. In Section 4, we expose the results we have obtained with new generated datasets and their variability. Finally, the conclusions and future work are presented in Section 5.

2 Ontology matching evaluation and test generation

In this section, we briefly present the current setting of ontology matching evaluation (Section 2.1), the Benchmark data set (Section 2.2) and the state-of-the-art in alignment test generators (Section 2.3). The interested reader can find a broader overview of ontology matching evaluation in [4].

2.1 Evaluating ontology matching systems

Matching can be seen as an operation which takes as input two ontologies (o and o'), a set of parameters (p), a possibly empty partial alignment (A') and a set of resources (r) and outputs an alignment (A) between these ontologies (Fig. 1).

An alignment can be defined as a set of correspondences. A correspondence between two ontologies o and o' is a triple $\langle e, r, e' \rangle$, where e is an entity belonging to the first ontology, e' is an entity belonging to the second ontology, r is a relation, e.g., equivalence or subsumption, between them.

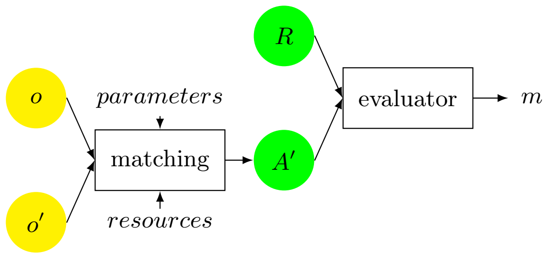

Fig. 1: Ontology matching process and evaluation (from [5]).

A matcher can be evaluated comparing its output alignment (A) with a reference alignment (R) using some measure (Fig. 1). Usually, such measures are precision, recall and F-measure [5]. Thus, in order to evaluate a matching system, one has to generate datasets in which a test is composed of two ontologies to be matched (o and o') and a reference alignment (R).

2.2 The Benchmark dataset

Benchmark aims at testing the strengths and the weaknesses of matching systems, depending on the availability of ontology features. This dataset has 111 tests, requiring to match an ontology written in OWL-DL to another one:

- Tests 1xx - compare the original ontology with itself, a random one and its generalization in OWL-Lite.
- Tests 2xx - compare the original ontology with the ontology obtained by applying the following set of modifications to it (Fig. 2):
  - names (naming conventions: synonyms, random strings, different generalization, translation into other language)
  - comments (no comments)
  - hierarchy (flattened hierarchy / expanded hierarchy / no specialization)
  - instances (no instance)
  - properties (no properties, no restrictions)
  - classes (flattened classes / expanded classes)
- Test 3xx - compare the original ontology with real ones found on the web.

Since 2004, Benchmark has been generated from the same seed ontology through the same set of XSLT stylesheets. This means, in particular, that no random modification is applied to these ontologies: the same 20% of classes are renamed and this renaming is always the same. This has advantages for studying the evolution of the field, because the test is always the same.

However, the Benchmark data set can be criticised on three main aspects:

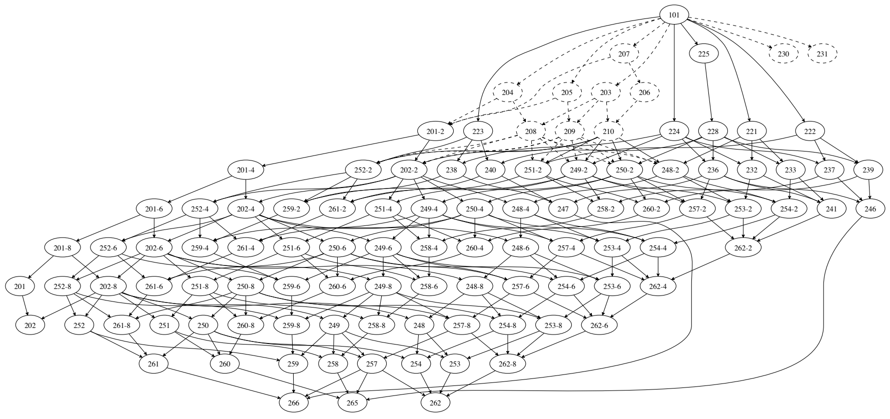

Fig. 2: The Benchmark lattice – the higher the test is in the hierarchy, the easier it is. Tests in dashed lines are not reproduced in the tests we used here (see §4).

Lack of realism Benchmark is not realistic because it covers a whole systematic space of mechanical alterations and in reality a matcher is not faced with such a space.

Lack in variability Benchmark always produces the same data set hence it is not variable. This covers three slightly different kinds of problems: (a) it can only be used with one seed ontology, (b) it always applies the same transformations (to the same entities), instead of applying them randomly, and (c) it is not flexible in the sense that it is not possible to produce an arbitrary test (such as 12% renaming, 64% discarding properties).

Lack of discriminability [7] Benchmark seems in general easy enough to OAEI participants so that they do not really allow them to make progress and to compare them. This is because, many of the proposed tests are easy and only a few are really difficult.

Our goal is to address variability and discriminability by producing a test generator (a) independent from the seed ontology, (b) with random modifications, and (c) which allows to fine tune parameters in order to cover the alteration space with any precision. With such a test generator it would be possible to generate different tests than Benchmark focusing on particular application profiles or particularly difficult cases.

We do not address the lack of realism because Benchmark has been designed to cover the problem space and not to offer one realistic profile[2]. Other initiatives, such as other tracks of OAEI and other generators, address this issue.

[2] One reviewer argues that we currently consider an alteration space, instead of a problem space, which assumes some realism, i.e., that these problems actually occurs. Moreover, (s)he write that we choose the kind of alteration and the granularity. This is right. But this alteration space is our attempt to cover, and not to represent, the problem space.

2.3 Other ontology alignment generators

So far, some alignment test generators have been developed.

An ontology generator inspired by Benchmark is the one developed in [2]. Its seed ontology is a random tree which is computed using a Gaussian distribution with average 4 and deviation 4 in order to determine the number of children per node. The second ontology is obtained from the first one by applying a set of alterations, similar to the ones used in Benchmark, such as label replacement, word addition or removal in labels, node deletion and node child addition and children shuffling. Then, these two generated ontologies are used to generate alignments between them. The aim of generating the original ontology is to perform realistic tests and to allow a wider coverage of variations in their structure.

The generator proposed in [10] satisfies two requirements: (a) to generate the structure and the instances of two taxonomies, and (b) to generate the mappings between these two generated taxonomies. Both taxonomies must have a fixed size and a Boltzmann sampler is used to achieve this. The probabilistic model used ensures an equal probability of appearance of a tree having a given size. Therefore, the input data is controlled using this sampler. The number of child nodes is controlled as well. Then, the mappings between the two taxonomies are generated, which must not be contradicted by the generated data. To achieve this goal, three constraints were enforced: the mappings must not introduce a cycle in the newly obtained graph (the mappings and the two given taxonomies), the mappings must not contradict the knowledge of the two taxonomies and they must not entail each other. In the end, instances are generated.

The TaxMe method [7] is build from existing directories and only approximates the reference alignment, it is not really a generator. In XML schema matching, STBenchmark [1] offers a way to generate one precise test (pair of schemas) by altering a source schema based on the combination of 11 base alterators. Their combination is defined through a set of input parameters. Swing [6] takes a similar approach as Benchmark and introduces a new interesting way of altering ontologies by using patterns. However, it is not an automatic generator and it is aimed at generating instance data: the same ontology is, in the end, used for all tests.

We decided to rewrite our own generator because we wanted to reproduce Benchmark first. Tournaire's generator was not suited because he was aiming at realism; Besana's generator would have been useful but was not available.

3 A modular benchmark test generator

We developed a test generator in Java based on the Jena API[3]. We present the principles of the generator (Section 3.1) and the testing strategy (Section 3.2).

[3] `http://jena.sourceforge.net/ontology/index.html`

3.1 Generator principles

Test generator architecture. The basic principles of the test generator is that, from one ontology, it can generate an altered one and an alignment between these two ontologies. The generator can as well accept a generated ontology, that is useful for generating scalability tests.

Because the alterations may be applied in sequence, we designed an alterator module taking as input an ontology and an alignment between this ontology and the seed ontology. This module outputs an altered ontology and an alignment between this ontology and the seed one (Fig. 3).

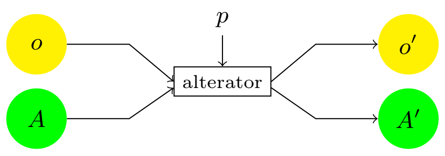

Fig. 3: Modular structure of test generators.

Test generator parameters. In order to assess the capability of matchers with respect to particular ontology features, we consider the following alterations: remove/add percentage of classes; remove/add percentage of properties; remove percentage of comments; remove percentage of restrictions; remove all classes from a level; rename percentage of classes; rename percentage of properties; add a number of classes to a specific level; flatten a level; remove individuals.

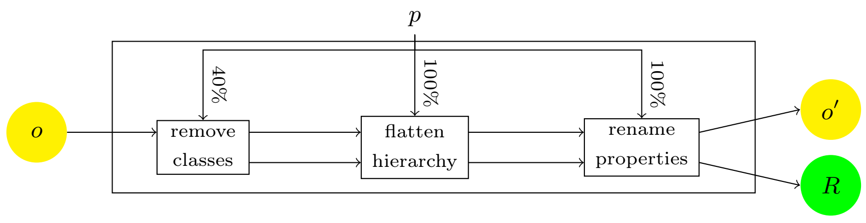

Fig. 4: Modular one-shot test generation.

Generating a dataset. For modifying an ontology according to a set of parameters we use the generator as illustrated in Fig. 4. It receives as input the seed ontology and the parameters which represent the alterations to be apply. The output is the modified ontology and the reference alignment. The program is implemented in a serial manner.

The test generator can be also used to reproduce data sets such as Benchmark. For that purpose, the program will either generate all the required tests independently by running in parallel the necessary composition of alterators

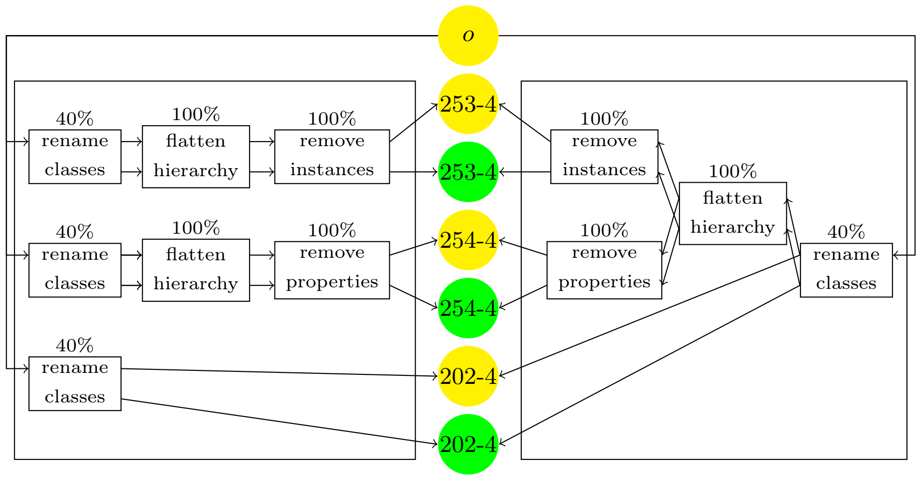

Fig. 5: Random (left) and continuous (left) test suite generation.

(Fig. 5, left) or generate them in sequence, as the initial Benchmark data set, i.e., by using a previous test and altering it further (Fig. 5, right). In the latter case, this corresponds to selecting paths in the lattice of Fig. 2 which cover the whole data set.

This approach may also be used to generate complete data sets covering the whole alteration space with a varying degree of precision (incrementing the alteration proportion by 50% or by 2%).

3.2 Preliminary experiments

Before evaluating matchers on the generated data sets, we have tested the generator through unit tests, checking if the percentage of alteration was indeed respected. Initially, the parameters were applied in a random order, using the bibliographic ontology as basis. From the results, we noticed a non expected matcher behaviour, that allowed us to improve the generation strategy.

Random vs. continuous policies. Contrary to expected, matchers did not had a continuous degradation of their performances as more alterations were applied. This made difficult to read one Benchmark test result, as developers would like to read them. This is the consequence of generating the dataset fully randomly (Fig. 5, left), where each test is generated independently from the others. In this modality, some tests with more alterations may be easier than other with less alterations by chance.

We validated this explanation by generating continuous tests (Fig. 5, right) as Benchmark were generated. In this case, new tests are generated from previous ones with the modular architecture of the generator. This behaviour is only observable locally, i.e., on one data set. When generating randomly several datasets, matcher behaviours are on average continuous. In results reported below, half of the tests were obtained with the random generation and half of them with continuous generation. Their results are the same (within 1 percentage point variation).

Modification dependencies. We observed that test difficulty may not be the same across tests supposed to have the same amount of alteration. This is explained by the dependency between alterations. Consider, for example, a scenario in which we would like to remove 60% of classes and to rename 20% of classes. According to these two parameters, three extreme cases may happen (as illustrated in Fig. 6):

- rename 20% of classes and then remove 60% of classes, including all renamed classes. In this situation, the test is easier than expected because all renamed classes have been removed;
- rename 20% of classes and then remove 60% of classes, including a part of renamed classes. In this situation, the test is as hard as expected because the required proportion of the renamed classes have been removed.
- rename 20% of classes and then remove 60% of classes, without removing a renamed class. In this situation, the test is harder than expected because none of the renamed classes has been removed.

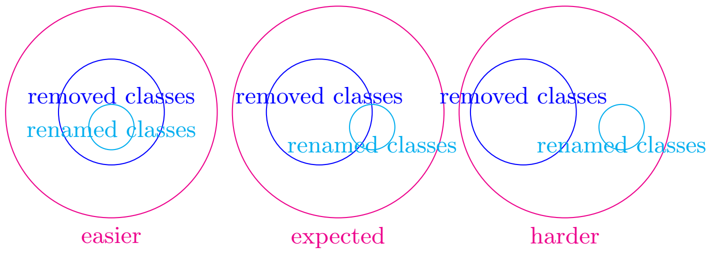

Fig. 6: Test dependency.

Hence, a random disposition of parameters might reduce the really hard cases. As can be seen from the example, the nominal expected case may be restored by removing 60% of the classes before renaming 20% of the remaining. Therefore, we established a relevant order for parameters: remove classes, remove properties, remove comments, remove restrictions, add classes, add properties, rename classes, rename properties. In this way, we obtained the expected results. This order helps determining the paths in Fig. 2 used for generating Benchmark. Such an order was not previously observed in the Benchmark tests because the value of parameters, except rename resources, was set to the value of 100%.

4 Benchmark validity

In order to test the validity of the Benchmark dataset principles, we used the test generator to reproduce them with different characteristics. Three modalities were used in the evaluation:

1. Regenerate the dataset using the same bibliographic ontology (biblio).
2. Generate datasets from two conference ontologies [11] (cmt and ekaw), of similar size and expressiveness as biblio.
3. Generate datasets from other ontologies of two other domains (tourism[4] and finance[5]).

These modalities were chosen because we assume that since participants have had the opportunity to test their systems with the original Benchmark, their results may be higher. The same holds for the conference dataset that these systems had the occasion to deal with, while the third group of tests is new for them. We thus expected them to be harder. We could evaluate the robustness of our generator, since the finance ontology has more than 300 classes and almost 2000 individuals.

In order to remove the possibility that the obtained results are an artifact of the generated test, we ran the tests five times for each method (continuous and random) and then we computed the average among the obtained results. Likewise, the tests are the same at each run (these tests are 201-202, 221-225, 228, 232-233, 236-241, 246-254, 257-262, 265-266). We decided not to reproduce the ones in which the labels are translated into another language or the ones in which the labels are replaced with their synonyms, because the corresponding alterators are not sufficiently good. The same algorithms with the same parameters have been used for all tests (the tests with the original Benchmark have been run again). In total, the results of this section are based on the execution of (1+(5 ontologies $\times$ 5 runs $\times$ 2 modalities) $\times$ 102 tests $\times$ 3 matchers =) 15606 matching tasks, i.e., a matcher has been run against a pair of ontologies and the result has been evaluated against a reference alignment.

4.1 Matchers

To test the generator, we used three different matchers participating in previous OAEI: Aroma, Anchor-Flood (Aflood) and Falcon-AO (Falcon). They are stable enough and generally available, yet sufficiently different.

Anchor-Flood tries to find alignments starting with some anchors and using the locality of a reference (super-concepts, sub-concepts, siblings, etc.) [8]. It is composed of two modules. The first one uses lexical and statistical information to extract the initial anchors. Then, starting with these anchors, it builds small blocks across ontologies and establishes the similarities between the two found blocks. Each similarity is stored in a partial alignment and, the process continues using as anchors the pairs found in the partial alignment. The process finishes when no more correspondences are found. We have used the preliminary OAEI 2010 version of Aflood.

Aroma [3] is divided in three stages. First, it builds a set of relevant terms for each entity, i.e. class or property. In order to achieve this, a single and binary

[4] http://www.bltk.ru/OWL/tourism.owl
[5] http://www.fadyart.com/ontologies/data/Finance.owl

term extractor applied to stemmed text is used to extract the vocabulary of a class or a property. In the second stage, the subsumption relations between entities are found using the implication intensity measure and an association rule model. Third, it establishes the best correspondence for each entity deducing first the equivalence relations, then suppressing the alignment graph cycles and the redundant correspondences. We have used Aroma 1.1.

Falcon-AO [9] is composed of two matchers: a linguistic (LMO) and a structural matcher (GMO). The linguistic matcher has two parts. The first one uses string comparisons and the second one uses virtual documents to describe each ontology entity. A virtual document is a "bag of words" containing the name, the comments, the labels and also neighbors names or labels of an entity. Vector space techniques are employed to measure the similarity between these virtual documents. The structural matcher represents an ontology as a bipartite graph and tries to find the similarity between the two input ontologies. In the end, if the result returned by the linguistic matcher is satisfying, it is returned. Otherwise, the structural matcher result is returned. We have used Falcon-AO 0.3.

4.2 Results

Table 1 provides the aggregated precision, recall and F-measure for each matcher and each data set. We discuss them only from the standpoint of F-measure because it allows for a more direct comparison.

	original			biblio			cmt			ekaw			tourism			finance			Σ
	P	F	R	P	F	R	P	F	R	P	F	R	P	F	R	P	F	R	F
Aflood	.99	**.87**	.78	.75	**.67**	.59	.95	**.72**	.58	.95	**.72**	.58	.94	**.76**	.62	.96	**.78**	.66	**.75**
Aroma	.79	**.63**	.53	.86	**.68**	.55	.92	**.65**	.50	.96	**.68**	.52	.85	**.74**	.64	.94	**.73**	.60	**.68**
Falcon	.83	**.76**	.70	.85	**.77**	.70	.82	**.69**	.59	.89	**.70**	.57	.83	**.73**	.65	.92	**.77**	.66	**.74**
Average		**.74**			**.71**			**.69**	.		**.70**			**.74**			**.76**		**72**

Table 1: Average results (on 5 random and 5 continuous runs) for the 2xx Benchmark series for the three matchers and six data sets.

First, we observed independently that the difficulties encountered by a matcher at a test are preserved across the seed ontology. Hence, Benchmark is useful for identifying weaknesses in matchers. What we mean here, is that by looking at the results, test by test, the relative performance of test for a matcher is preserved across seed ontologies. This is not visible in Table 1.

Then, for most of the matchers, the results obtained with the original Benchmark are not the highest. This goes counter our expectation of a positive bias in favour of the original Benchmark. In fact, the results are contrasted since Aflood has a significantly better score than with the reproduced Benchmark, Aroma has its worse score and Falcon has a very close score. It seems that the original

Benchmark is quite hard because for two matchers, results are better on the (randomised) reproduced Benchmark. Original Benchmark results, even if they do not show a systematic positive bias, seems to be the outlier.

The two other groups of tests, ekaw and cmt on one hand and tourism and finance on the other hand, have homogeneous results within the group and different results across groups. This indicates that the type of seed ontology has an influence on the results but for ontologies of the same type results are homogeneous. It seems that biblio is harder than conference which is harder than tourism-finance and this for all matchers (but Falcon).

But overall, results are found in the same range. If we exclude the results of Aflood and Aroma on the original Benchmark, the results of matchers vary of 11, 9 and 8 percentage points respectively.

Similarly, the order between matchers observed with the original Benchmark seems to be preserved in four out of six data sets. Surprisingly, the main outlier is the reproduced Benchmark. However, the few percentage point difference that is observed do not allow us to conclude, especially with respect to the 24 percentage points observed for the original Benchmark and the 10 percentage points in reproduced Benchmark.

What we observe is a relative homogeneity of these results: there is no more diversity across matchers than across data sets. In fact, the lack of discriminability observed in Benchmark is even reinforced in the other tests: the original Benchmark has 24 percentage points span while the reproduced biblio has only 10 points and the other data sets are lower. Hence, this is a property of the alterations generating Benchmark, thus another type of generation should be used.

5 Conclusion

In this paper we have looked for improving the tools available for evaluating ontology matchers. For that purpose, we have developed a test generator which follows a simple modular architecture and API. This generator does not depend on the seed ontology. It allows different modifications at each run of the program and the set of input parameters can be adjusted in order to cover the problem space with any precision. The generator can be extended by adding new ontology modifier modules and it can be used for generating individual tests with controlled characteristics as well as full data sets. Thus, we have largely improved the variability of generated tests.

The test generator was used to reproduce the Benchmark dataset not only for the bibliographic ontology, but for other ontologies with different structures. We observed that the obtained results are not better on original Benchmark than on new and different ontologies. This contradicts the assumption that there is a systematic positive bias towards Benchmark.

We observed that for the same type of seed ontology, each matcher has homogeneous results and that the order between matchers obtained on the original

Benchmark (Aflood, Falcon, Aroma) was preserved in four cases out of six. However, the difference between systems is too small to draw definitive conclusions. The new tests still lack discriminability. It is thus a feature of the Benchmark generation modalities.

We plan to improve these experiments by using other matching systems. This can be achieved using the SEALS platform and it is planned for the OAEI 2011 campaign for which the generator will be used. We may also use different more difficult generation modalities, in order to increase discriminability. Another perspective is to use the test generator for exploring the notion of test hardness, that could help to better approach the lack of discriminability.

Acknowledgements

This work has been partially supported by the SEALS project (IST-2009-238975).

References

1. Bogdan Alexe, Wang-Chiew Tan, and Yannis Velegrakis. STBenchmark: towards a benchmark for mapping systems. In *Proc. 34th Very Large Databases conference (VLDB), Auckland (NZ)*, pages 230–244, 2008.
2. Paolo Besana. *Predicting the content of peer-to-peer interactions*. PhD thesis, University of Edinburgh, 2009.
3. Jérôme David, Fabrice Guillet, and Henri Briand. Association rule ontology matching approach. *International Journal of Semantic Web and Information Systems*, 3(2):27–49, 2007.
4. Jérôme Euzenat, Christian Meilicke, Heiner Stuckenschmidt, Pavel Shvaiko, and Cássia Trojahn dos Santos. Ontology alignment evaluation initiative: Six years of experience. *Journal of Data Semantics*, XV:158–192, 2011.
5. Jérôme Euzenat and Pavel Shvaiko. *Ontology matching*. Springer-Verlag, Heidelberg (DE), 2007.
6. Alfio Ferrara, Stefano Montanelli, Jan Noessner, and Heiner Stuckenschmidt. Benchmarking matching applications on the semantic web. In *Proc. 8th Extended Semantic Web Conference (ESWC), Herssounisos (GR)*, number 6644 in Lecture notes in computer science, pages 108–122, 2011.
7. Fausto Giunchiglia, Mikalai Yatskevich, Paolo Avesani, and Pavel Shvaiko. A large scale dataset for the evaluation of ontology matching systems. *Knowledge engineering review*, 24(2):137–157, 2009.
8. Md. Seddiqui Hanif and Masaki Aono. Anchor-flood: Results for OAEI 2009. In *Proc. 4th ISWC workshop on ontology matching (OM), Washington (DC US)*, 2009.
9. Wei Hu, Yuzhong Qu, and Gong Cheng. Matching large ontologies: A divide-and-conquer approach. *Data and Knowledge Engineering*, 67(1):140–160, 2008.
10. Rémi Tournaire. *Découverte automatique de correspondances entre ontologies*. PhD thesis, Université de Grenoble, 2010.
11. Ondřej Šváb, Vojtěch Svátek, Petr Berka, Dušan Rak, and Petr Tomášek. Ontofarm: Towards an experimental collection of parallel ontologies. In *Poster Track of ISWC*, 2005.