How well does your Instance Matching system perform? Experimental evaluation with LANCE

Tzanina Saveta[1], Evangelia Daskalaki[1], Giorgos Flouris[1], Irini Fundulaki[1], and Axel-Cyrille Ngonga Ngomo[2]

[1] Institute of Computer Science-FORTH* Greece,
[2] IFI/AKSW, University of Leipzig, Germany

Abstract. Identifying duplicate instances in the Data Web is most commonly performed (semi-)automatically using instance matching frameworks. However, current instance matching benchmarks fail to provide end users and developers with the necessary insights pertaining to how current frameworks behave when dealing with real data. In this paper, we present the results of the evaluation of instance matching systems using LANCE, a domain-independent, *schema agnostic* instance matching benchmark generator for Linked Data. LANCE is the first benchmark generator for Linked Data to support *semantics-aware* test cases that take into account complex OWL constructs in addition to the standard test cases related to structure and value transformations. We provide a comparative analysis with benchmarks produced using the LANCE framework for different domains to assess and identify the capabilities of state of the art instance matching systems.

1 Introduction

Instance matching (IM), refers to the problem of identifying instances that describe the *same real-world object*. With the increasing adoption of Semantic Web technologies and the publication of large interrelated RDF datasets and ontologies that form the Linked Data (LD) Cloud, a number of IM techniques adapted to this setting have been proposed [1,2,3].

Clearly, the large variety of IM techniques requires their comparative evaluation to determine which technique is best suited for a given application. Assessing the performance of these systems generally requires *well-defined and widely accepted benchmarks* to allow determining the weak and strong points of the methods or systems, as well as for motivating the development of better systems to overcome the identified weak points. Hence, properly designed benchmarks help push the limit of existing systems [4,5,6,7,8], advancing both research and technology.

Recently LANCE [8], a state-of-the-art benchmark generator for benchmarking instance matching systems in the LD context was introduced. LANCE is a flexible, generic, domain-independent and schema-agnostic benchmark generator for IM systems. LANCE supports a large variety of value, structure based and semantics-aware transformations with varying degrees of difficulty. The results

* The presented work was funded by the H2020 project HOBBIT (#688227).

of these transformations are recorded in the form of a weighted gold standard that allows a more fine-grained analysis of the performance of instance matching tools. Details on the different transformation types, our weighted gold standard and metrics, as well as the evaluation of our system can be found in [8].

In the current paper, our focus lies on evaluating state-of-the-art instance matching systems with benchmarks produced using the LANCE framework. The purpose of this evaluation is to provide further insights on the weak and strong points of different IM systems, that would be complementary to the ones already established in [8]. In particular, we evaluate the effect of using different datasets as input to the benchmark generator module of LANCE, and show that the performance of IM systems is not only affected by the benchmark creation process itself, but also by the characteristics of the input dataset that was used to generate the benchmark. For our tests, we used SPIMBENCH [7] and UOBM [9] datasets.

2 LANCE Approach

Here, we give the basic features of LANCE. The interested reader can find more details in [8]:

Transformation-based test cases. LANCE supports a set of *test cases* based on *transformations* that distinguish different types of matching entities. Similarly to existing IM benchmarks, LANCE supports *value-based* (typos, date/number formats, etc.) and *structure-based* (deletion of classes/properties, aggregations, splits, etc.) test cases. LANCE is the *first benchmark generator* to support *semantics-aware* test cases that go beyond the standard RDFS constructs and allow testing the ability of IM systems to use the semantics of RDFS/OWL axioms to identify matches and include tests involving *instance (in)equality*, class and property *equivalence* and *disjointness*, *property constraints*, as well as *complex class definitions*. LANCE also supports *simple combination (SC)* test cases (implemented using the aforementioned transformations applied on different triples pertaining to the same instance), as well as *complex combination (CC)* test cases (implemented by combinations of individual transformations on the same triple).

Similarity score and fine-grained evaluation metrics. LANCE provides an enriched, *weighted gold standard* and related evaluation metrics, which allow a more fine-grained analysis of the performance of systems for tests with varying difficulty. The gold standard indicates the matches between source and target instances. In particular, each match in the gold standard is enriched with annotations specific to the test case that generated each pair, i.e., the type of test case it represents, the property on which a transformation was applied, and a *similarity score* (or *weight*) of the pair of reported matched instances that essentially quantifies the difficulty of finding a particular match. This detailed information allows LANCE to provide more detailed views and novel evaluation metrics to assess the completeness, soundness, and overall matching quality of an IM system on top of the standard precision/recall metrics. Thus, LANCE provides fine-grained information to support debugging and extending IM systems.

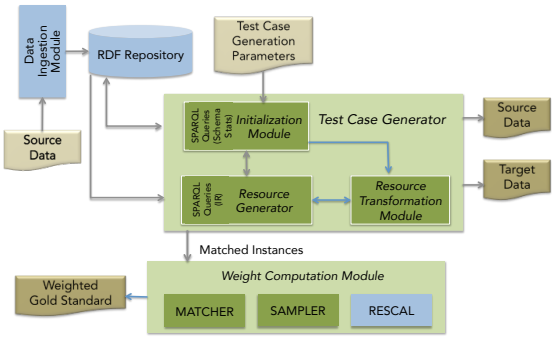

Fig. 1. LANCE System Architecture

High level of customization LANCE provides the ability to build benchmarks with different characteristics on top of any input dataset, thereby allowing the implementation of diverse test cases for different domains, dataset sizes and morphology. This makes LANCE highly customizable and domain independent;
Implementation of LANCE. LANCE[1] is a highly configurable instance matching benchmark generator for Linked Data that consists of two components : (i) an *RDF repository* that stores the *source datasets* and (ii) a *test case generator* (see Figure 1). The test case generator takes as input a *source* dataset and produces a *target* dataset that implements various *test cases* according to the specified configuration parameters to be used for testing instance matching tools. It consists of the *Initialization*, the *Resource Generator* and the *Resource Transformation* modules.

- *Initialization module* reads the test case generation parameters and retrieves by means of SPARQL queries the schema information (e.g., schema classes and properties) from the RDF repository that will be used for producing the target dataset.
- The *Resource Generator* uses this input to retrieve instances of those schema constructs from the RDF repository and passes those (along with the configuration parameters) to the *Resource Transformation Module*.
- The *Resource Transformation* module returns for a source instance u_i the transformed instance u_i' and stores this in the target dataset; this module is also responsible in producing an entry in the gold standard. Once LANCE has performed all the requested transformations, the *Weight Computation Module* calculates the similarity scores of the produced matches. The configuration parameters specify the part of the schema and data to consider when producing the different test cases as well as the percentage and type of transformations to consider. More specifically, parameters for *value-based* test cases specify the kind and severity of transformations to be applied; for *structure* and *semantics-aware* test cases, the parameters specify the type of transformations to be considered. The idea behind configuration parameters is to allow one to tune the benchmark generator into producing benchmarks

[1] The code of LANCE is available at `https://github.com/jsaveta/Lance`

of varying degrees of difficulty which test different aspects of an instance matching tool.

LANCE is implemented in Java and in the current version we use OWLIM Version 2.7.3. as our RDF repository.

3 Experimental Results

Settings. Our evaluation focused on demonstrating the capability of our benchmark generator in assessing and identifying the strengths and weaknesses of instance matching systems. For this purpose, we evaluated LogMap Version 2.4 [10] using the MoRe [11] reasoner, OtO [12] and LIMES [2] running the EAGLE [13] algorithm. We chose these tools because they are prototypical working instances of existing IM systems. Attempts to evaluate systems such as RiMOM-IM [14], COMA++ [15] and CODI [16] with LANCE were not successful due to issues from the systems' side. We were not able to work with RiMOM-IM due to incomplete information regarding the use of the system; COMA++ supports instance-based ontology matching but does not aim for instance matching per se. CODI is no longer supported by their development team. LogMap considers both schema and instance level matching; OtO on the other hand, needs to be configured manually to implement instance matching tasks. The same holds for EAGLE, which can learn specifications and focuses on instance matching tasks only. In order to identify strong and weak points of state-of-the-art IM systems, we tested the tools at hand with difficult tasks in which we transform the entirety of the source dataset to produce the target dataset. All experiments were conducted on an Intel(R) Core(TM) 2 Duo CPU E8400 @3.00GHz with 8G of main memory running Windows 7 (64-bit).

Datasets. We used as source datasets produced by LDBC's[2] SPIMBENCH [7] and UOBM's [9] data generators. SPIMBENCH datasets are described using a rich ontology with many different OWL constructs, in contrast with UOBM that employs a simpler ontology with many object and some datatype properties. For each generator (SPIMBENCH, UOBM) we produced two datasets, one with 10K triples and one with 50K triples. For SPIMBENCH those triples approximately correspond to 500 and 2.5K instances respectively and for UOBM to 2K and 10K.

Results. Figures 2 and 3 report the results for the different types of test cases and for the aforementioned datasets. In all cases, we measured recall, precision and f-measure, along with the *similarity score* and *standard deviation*.

Regarding the SPIMBENCH dataset, LogMap responds well to the value-based test cases having a high precision and recall (close to 0.75) but its performance degrades when the instances are involved in semantics-aware test cases giving low precision and recall (below 0.4). Despite of these results we claim that LogMap performs sufficiently well when faced with semantics-aware transformations since it is called to perform a matching task for highly heterogeneous datasets. OtO gives very good precision results for the value-based test cases but in some cases it is not able to find any match (recall is below 0.1).

[2] LDBC Semantic Publishing Benchmark: `http://ldbcouncil.org/developer/spb`

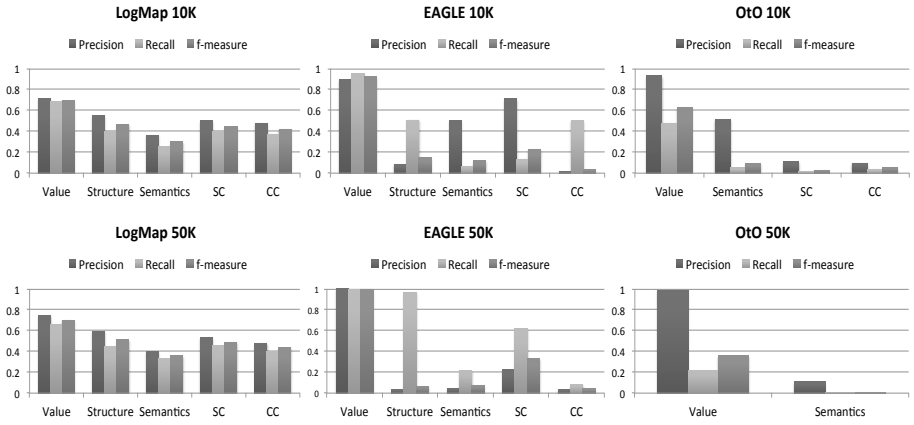

Fig. 2. Applicability experiments for LogMap, EAGLE and OtO for SPIMBENCH

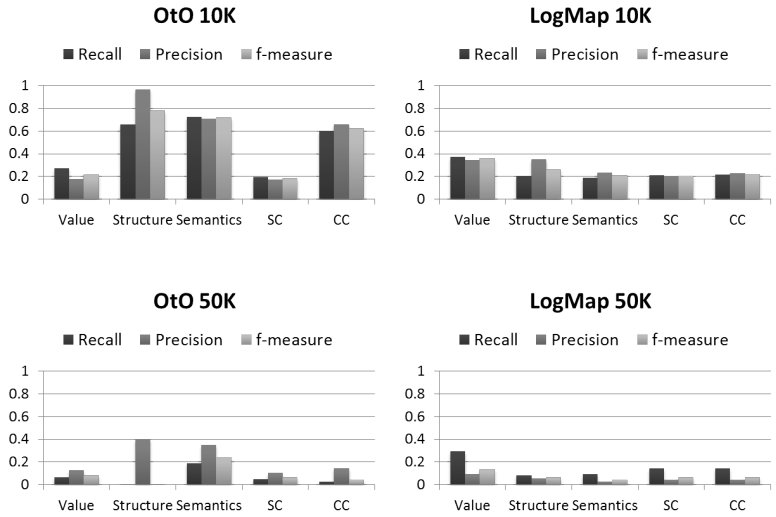

Fig. 3. Applicability experiments for LogMap and OtO for UOBM

The algorithm of EAGLE performs well when faced with syntactic transformations. Increasing changes to the topology of the underlying RDF graphs (the case of semantics-aware test cases) leads to a degradation of the performance of the algorithm. The performance of EAGLE is not consistent since it is non-deterministic and uses unsupervised learning.

The second experiment that we conducted compared the similarity scores as well as the standard deviation of the results of the systems with those of LANCE when the latter is used as a baseline. These metrics provide insights on the

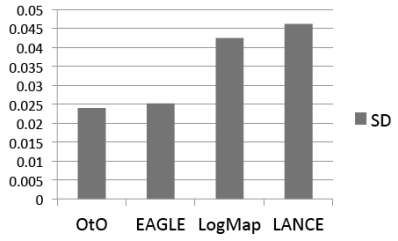

Fig. 4. Standard Deviation for LogMap, EAGLE, OtO, for 10K and semantics-aware test cases for SPIMBENCH.

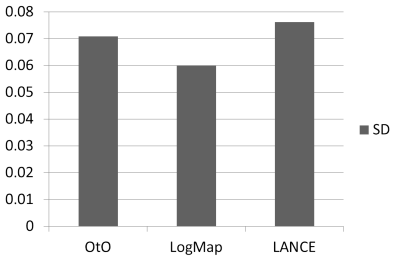

Fig. 5. Standard Deviation for LogMap and OtO, for 10K and structure-based test cases for UOBM.

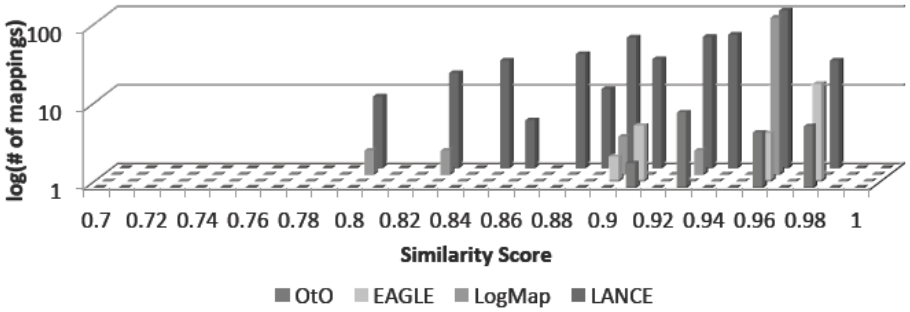

Fig. 6. Similarity score distribution for LogMap, EAGLE, OtO, for 10K and semantics-aware test cases for SPIMBENCH. The similarity score of the benchmark is also shown (column LANCE).

ability of the systems to address the challenges proposed by LANCE benchmarks. Figures 4 and 6 give the standard deviation and similarity scores for all three systems and for the semantics-aware test case for the 10K triples SPIMBENCH dataset. They also show the values for LANCE in order to have a baseline for comparison. We can see that LogMap reports scores and standard deviation close to the ones given by LANCE verifying that it can address the "difficult" test cases. EAGLE and OtO report lower similarity scores and standard deviation, meaning that they cannot address the challenges imposed by the, harder, semantics-aware test cases.

For UOBM, we ran LogMap and OtO, but not EAGLE because we were not able to correctly initialize it. UOBM datasets seem to be more "difficult" for both IM systems, and this difficulty stems from the dataset itself, rather than from the transformations imposed by LANCE. In particular, an important source of difficulty for the systems derives from the fact that the URIs of the instances in the dataset look very similar to each other, so even the change of a URI can lead to false positives or false negatives. To conclude, LogMap does not respond

well to any of the categories, but its performance is not affected by the dataset size. On the other hand, OtO responds better, but is affected negatively when the dataset gets larger. Figures 5 and 7 give the standard deviation and similarity scores for both systems and for the structure-based test case for the 10K triples UOBM dataset. We can also see that OtO reports scores and standard deviation results slightly closer to the ones given by LANCE than LogMap, verifying that it can address more difficult test cases.

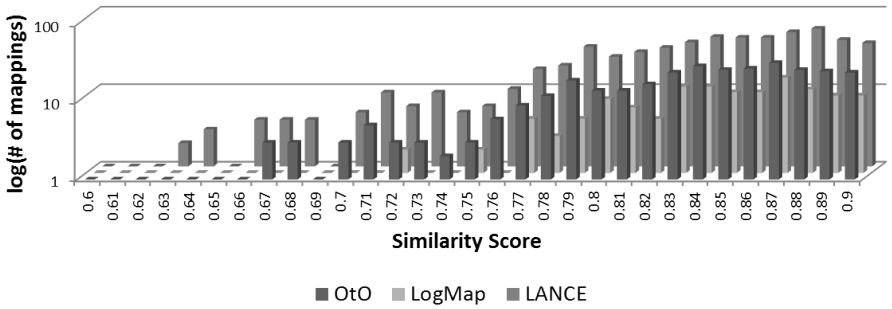

Fig. 7. Similarity score distribution for LogMap and OtO, for 10K and structure-based test cases for UOBM. The similarity score of the benchmark is also shown (column LANCE).

In our previous evaluation [8], we showed how the different types of transformations affected the performance of IM systems when tested using LANCE. The current work shows that the dataset used as a source for generating the benchmark is also an important factor that may affect such performance. This conclusion is derived by the fact that, even though we used the same parameters for the transformations in SPIMBENCH and UOBM and for all sizes, the systems did not respond similarly in the two datasets and dataset sizes as one might expect. This phenomenon was explained in Section 3, but note that our conclusions do not necessarily generalize to other datasets. This is a novel insight that we plan to exploit further by determining the dataset characteristics that are critical for this effect (e.g., dataset structure, URI format and scheme, dataset domain, etc.).

4 Conclusions

In our previous work we introduced LANCE, an instance matching benchmark generator focusing on benchmarking instance matching systems for Linked Data. LANCE is a domain-independent, highly modular and configurable generator that can accept as input *any* linked dataset and its accompanying schema to produce a target dataset implementing matching tasks of varying levels of difficulty.

In the current paper, we ran experiments which used benchmarks generated by LANCE to evaluate state-of-the-art IM systems. These experiments should be viewed as an addendum to the experiments appearing in [8], and have provided

additional insights on the factors that affect the performance of an IM system. In fact, it was shown that it is not only the types (and difficulty) of the transformations imposed by LANCE that affect a system's performance, but also the characteristics of the source dataset may play an important role.

In the future, we plan to study further this observation by pinpointing those characteristics of a dataset that have the most important effect on the systems' performance. Regarding LANCE itself, we will consider extensions for spatial and streaming data; we also intend to work with datasets that include *blank nodes* thereby creating more challenging tasks for instance matching tools. Furthermore, we plan to study the frequency of appearance of the various types of transformations in real datasets in order to be able to propose mixes of different transformations that are more realistic with respect to actual datasets.

References

1. R. Isele, A. Jentzsch, and C. Bizer. Silk Server - Adding missing Links while consuming Linked Data. In *COLD*, 2010.
2. A.-C. Ngonga Ngomo and Soren Auer. LIMES - A Time-Efficient Approach for Large-Scale Link Discovery on the Web of Data. *IJCAI*, 2011.
3. K. Stefanidis, V. Efthymiou, M. Herschel, and V. Christophides. Entity resolution in the web of data. In *WWW, Companion Volume*, 2014.
4. Ontology Alignment Evaluation Initiative. `http://oaei.ontologymatching.org/`.
5. K. Zaiss, S. Conrad, and S. Vater. A Benchmark for Testing Instance-Based Ontology Matching Methods. In *KMIS*, 2010.
6. B. Alexe, W.-C Tan, and Y. Velegrakis. STBenchmark: Towards a benchmark for mapping systems. In *PVLDB*, 2008.
7. T. Saveta, E. Daskalaki, G. Flouris, et al. Pushing the Limits of Instance Matching Systems: A Semantics-Aware Benchmark for Linked Data. In *WWW (Companion Volume)*, 2015.
8. T. Saveta, E. Daskalaki, G. Flouris, I. Fundulaki, and A.-C. Ngonga. LANCE: Piercing to the Heart of Instance Matching Tools. In *ISWC*, 2015.
9. L. Ma, Y. Yang, Z. Qiu, et al. Towards a Complete OWL Ontology Benchmark. In *ESWC*, 2006.
10. E. Jiménez-Ruiz and B. C. Grau. Logmap: Logic-based and scalable ontology matching. In *ISWC*, 2011.
11. A. A. Romero, B.C. Grau, I. Horrocks Ian, and E. Jiménez-Ruiz. MORe: a Modular OWL Reasoner for Ontology Classification. In *ORE*, 2013.
12. E. Daskalaki and D. Plexousakis. OtO Matching System: A Multi-strategy Approach to Instance Matching. In *CAiSE*, 2012.
13. A.-C. Ngonga Ngomo and K. Lyko. EAGLE: Efficient Active Learning of Link Specifications using Genetic Programming. In *ESWC*, 2012.
14. J. Li, J. Tang, Y. Li, and Q. Luo. Rimom: A dynamic multistrategy ontology alignment framework. *TKDE*, 21(8), 2009.
15. S. Massmann, S. Raunich, D. Aumüller, P. Arnold, and E. Rahm. Evolution of the COMA match system. *Ontology Matching*, 49, 2011.
16. J. Euzenat, A. Ferrara, W. R. van Hage, et al. Results of the ontology alignment evaluation initiative 2011. In *OM*, 2011.