

# SPIMBENCH: A Scalable, Schema-Aware Instance Matching Benchmark for the Semantic Publishing Domain

T. Saveta<sup>1</sup>, E. Daskalaki<sup>1</sup>, G. Flouris<sup>1</sup>, I. Fundulaki<sup>1</sup>, M. Herschel<sup>2</sup>, A.-C. Ngonga Ngomo<sup>3</sup>  
 {jsaveta, eva, fgeo, fundul}@ics.forth.gr, melanie.herschel@ipvs.uni-stuttgart.de, ngonga@informatik.uni-leipzig.de

#1 FORTH-ICS, #2 University of Stuttgart, #3 University of Leipzig



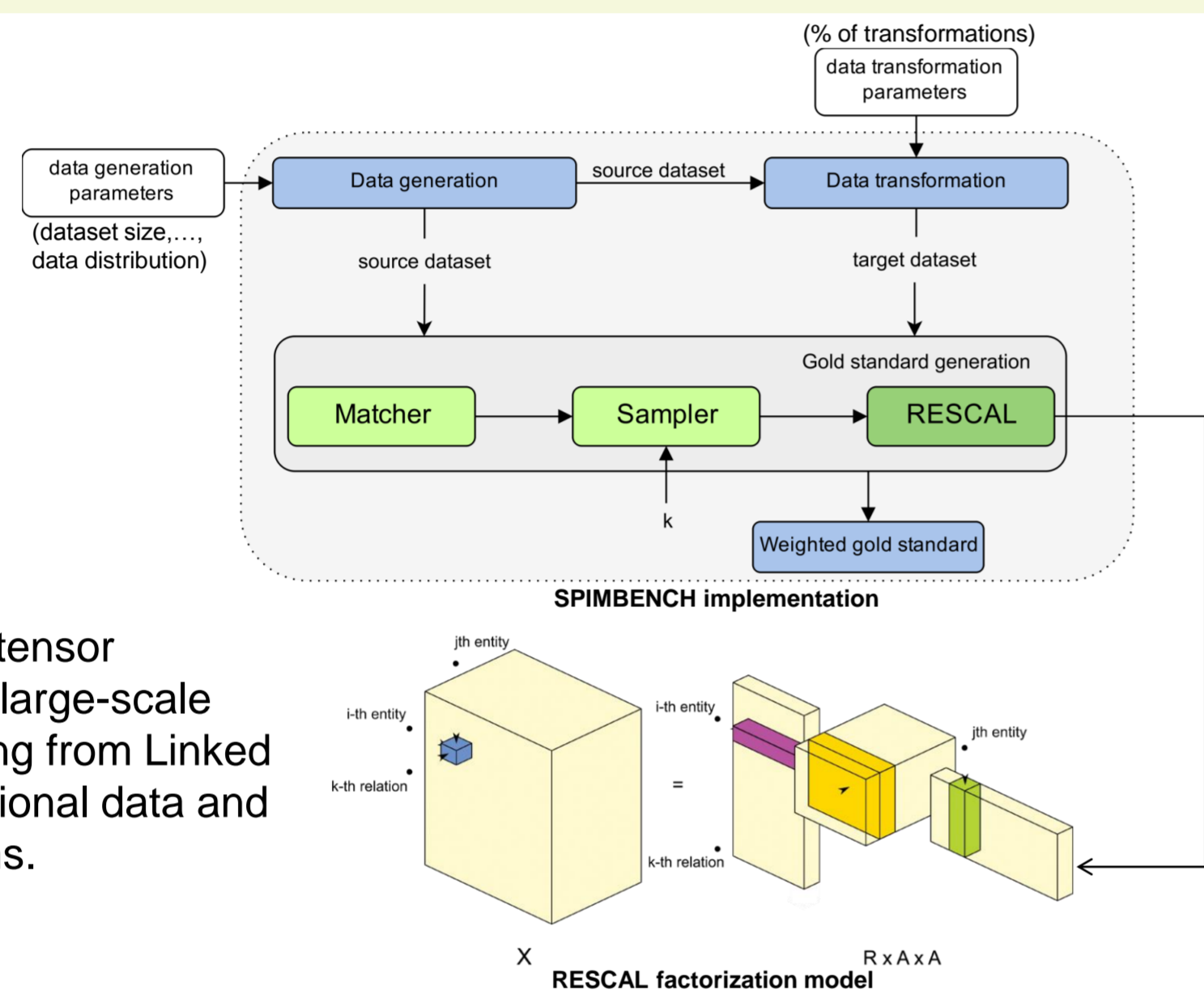
## Motivation

The widespread adoption of Semantic Web Technologies and the publication of large interrelated RDF datasets and ontologies in the Web has made the integration of data a crucial task. Data linking in this context is essential in order to provide an integrated view of the underlying information; this is achieved by instance and schema matching techniques. To aid the users to choose among the systems that perform such tasks, a number of benchmarks have been developed.

## SPIMBENCH Approach

**SPIMBENCH** is a benchmark for the Semantic Publishing Domain which takes into consideration RDFS and OWL constructs in order to evaluate instance matching systems. SPIMBENCH supports:

- A data generator that extends the one provided by LDBC's SPB Benchmark.
- Semantics aware transformations.
- Standard value and structure based transformations.<sup>[2,3]</sup>
- Scalable data generation in order of billion triples.
- Weighted gold standard based on tensor factorization.



**RESCAL**<sup>[1]</sup> is a tensor factorization for large-scale relational learning from Linked Data, multi-relational data and large multigraphs.

## Transformations

### Value-based

- Blank Character Addition/Deletion
- Random Character Addition/Deletion/Modification
- Token Addition/Deletion/Shuffle
- Date Format
- Abbreviation
- Synonym/Antonym
- Stem of a Word
- Multilinguality

### Structure-based

- Property Addition/Deletion
- Property Aggregation
- Property Extraction

### Semantics-aware

RDFS/OWL	SD	TD	SCHEMA TRIPLES	GS
owl:sameAs	$(u_1, \text{rdf:type}, C)$ $(u_2, \text{rdf:type}, C)$	$(u_1', \text{rdf:type}, C)$ $(u_2', \text{rdf:type}, C)$ $(u_1', \text{owl:sameAs}, u_2')$		$u_1 \sim u_1'$ $u_1 \sim u_2'$ $u_2 \sim u_2'$ $u_2 \sim u_1'$
owl:differentFrom	$(u_1, \text{rdf:type}, C)$	$(u_1', \text{rdf:type}, C)$ $(u_1'', \text{rdf:type}, C)$ $(u_1', \text{owl:differentFrom}, u_1'')$		$u_1 \sim u_1'$
owl:equivalentClass	$(u_1, \text{rdf:type}, C)$	$(u_1', \text{rdf:type}, C')$	$(C, \text{owl:equivalentClass}, C')$	$u_1 \sim u_1'$
owl:disjointWith	$(u_1, \text{rdf:type}, C)$	$(u_1', \text{rdf:type}, C')$	$(C, \text{owl:disjointWith}, C')$	
owl:FunctionalProperty	$(u_1, \text{rdf:type}, C)$ $(u_1, p_1, O_1)$	$(u_1', \text{rdf:type}, C)$ $(u_1', p_1, O_2)$	$(p_1, \text{rdf:type}, \text{owl:FunctionalProperty})$	$O_1 \sim O_2$
owl:InverseFunctionalProperty	$(u_1, \text{rdf:type}, C)$ $(u_1, p_1, O_1)$	$(u_1', \text{rdf:type}, C)$ $(O_1, p_1, u_1')$	$(p_1, \text{rdf:type}, \text{owl:InverseFunctionalProperty})$	$u_1 \sim u_1'$
owl:unionOf	$(u_1, \text{rdf:type}, C)$	$(u_1', \text{rdf:type}, C')$	$(C', \text{owl:unionOf}, \{C_0, C_1, \dots\})$	$u_1 \sim u_1'$
owl:intersectionOf	$(u_1, \text{rdf:type}, C)$	$(u_1', \text{rdf:type}, C')$	$C \text{ owl:intersectionOf } \{C, D, E, F\}$ $C' \text{ owl:intersectionOf } \{C, D\}$	$u_1 \sim u_1'$

## Combination of transformations

More than one transformation types per instance.

### Simple

One transformation per triple.

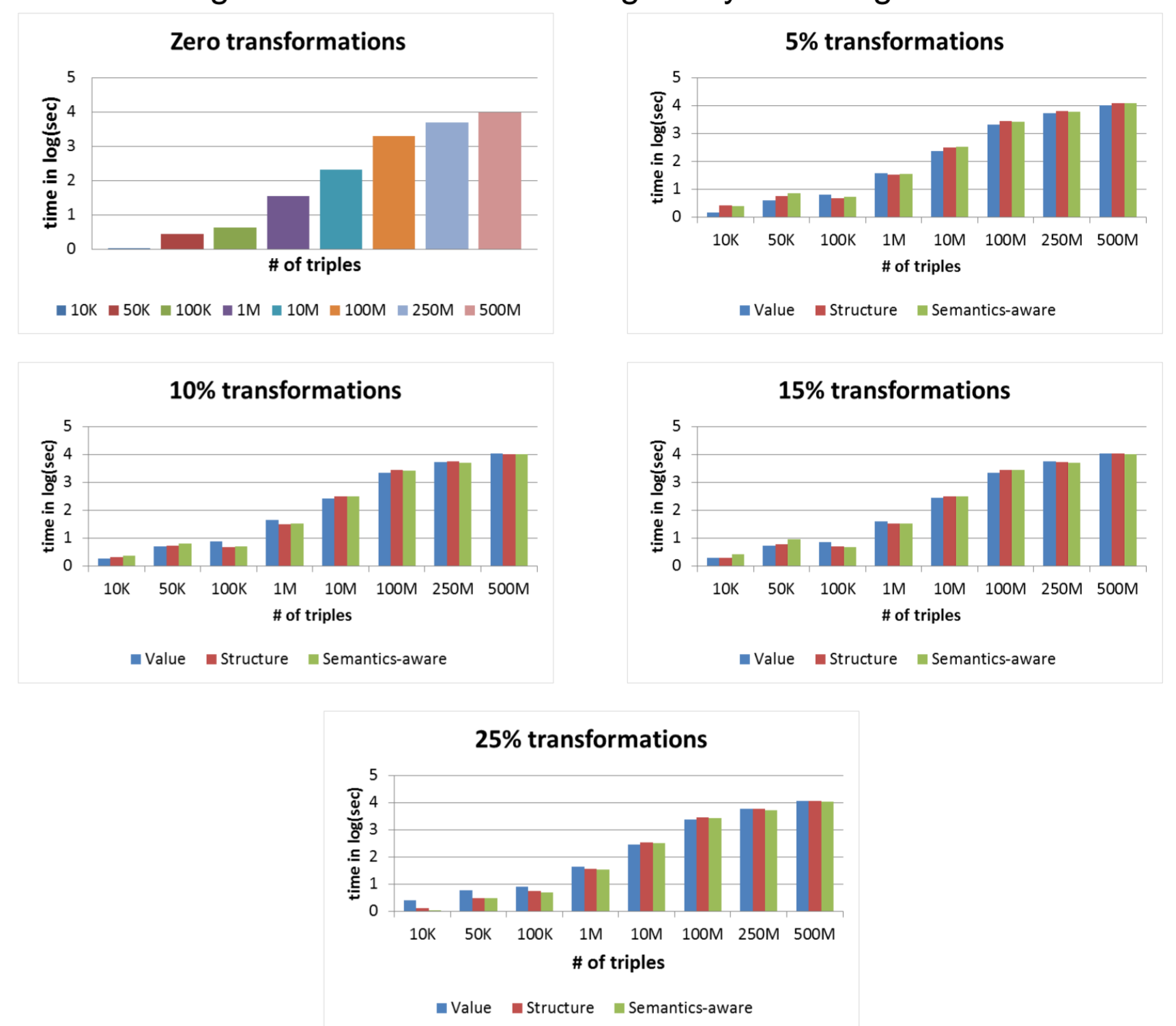
### Complex

Combination of two transformations per triple (value-based and structure-based or value-based and semantics-aware) based on the transformation parameters.

## Scalability

Scalability experiments for datasets up to 500M triples with simple combination of transformations.

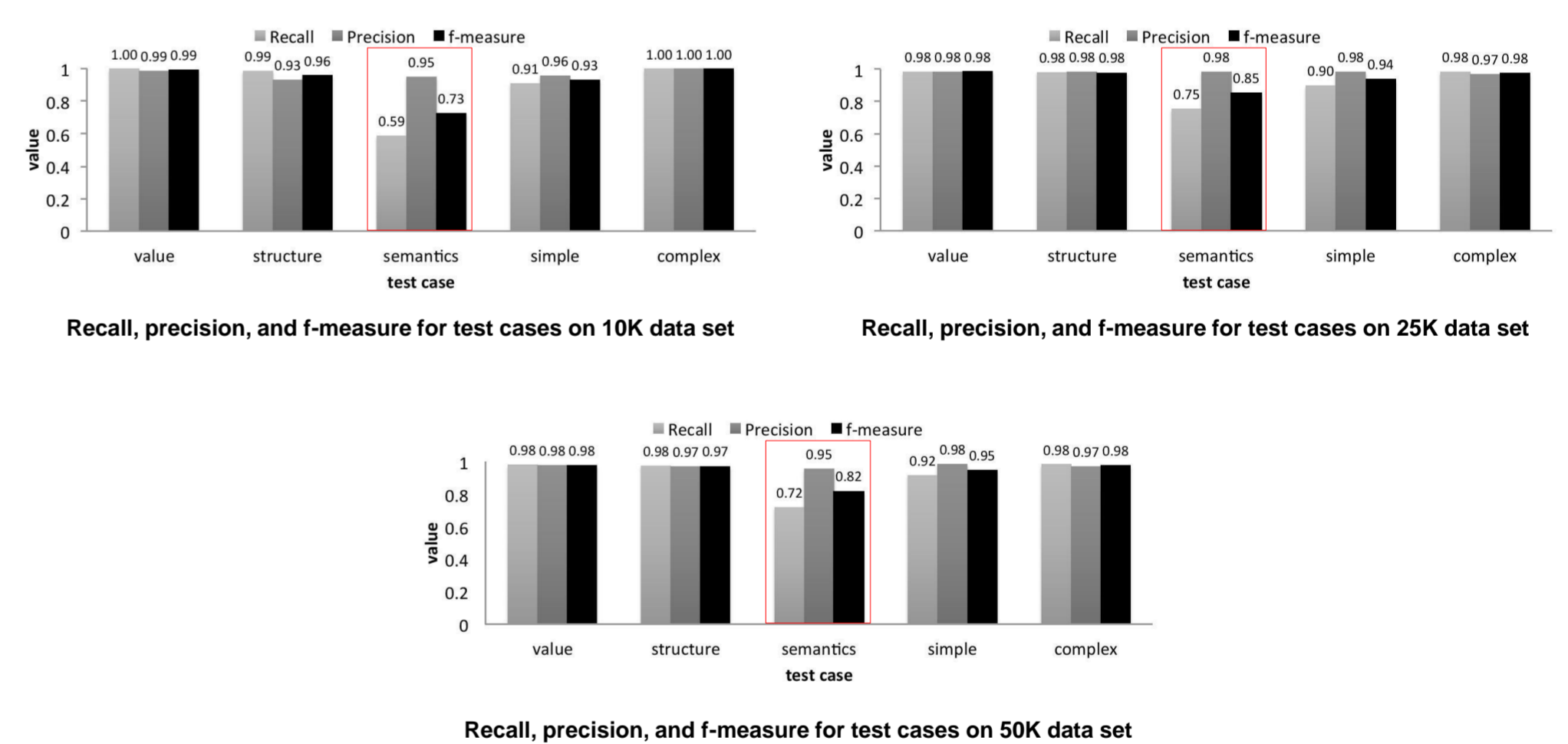
- 1000 triples ~ 36 entities.
- Data generation is linear to the size of triples.
- Transformation overhead is negligible for value, structure-based, semantics-aware and simple combinations.
- Overhead for logical transformations is higher by one magnitude.



Scalability experiments for n% of simple combination transformation type

## Applicability of SPIMBENCH

We demonstrated the applicability of SPIMBENCH by using it to evaluate LogMap<sup>[4]</sup> with different data set sizes and different test cases.



LogMap responds optimally regarding the precision as it does not find many matches that are not actually a match. On the other hand, fails to find matches when the instance is involved in multiple semantics-aware test cases.

## Future Work

- Domain independent instance matching test case generator for Linked Data.
- Definition of more sophisticated metrics that takes into account the difficulty (weight).

## Acknowledgments

This work was partially supported by the ongoing FP7 European Project LDBC (Linked Data Benchmark Council).

## References

- [1] Maximilian Nickel, and Volker Tresp. Tensor Factorization for Multi-relational Learning. ECML/PKDD 3, volume 8190 of Lecture Notes in Computer Science, page 617-621. Springer, 2013.
- [2] A. Ferrara, D. Lorusso, S. Montanelli, and G. Varese. Towards a Benchmark for Instance Matching. In OM, 2008.
- [3] A. Ferrara, S. Montanelli, J. Noessner, and H. Stuckenschmidt. Benchmarking Matching Applications on the Semantic Web. In ESWC, 2011.
- [4] E. Jimenez-Ruiz and B. C. Grau. Logmap: Logic-based and scalable ontology matching. In ISWC, 2011.



The research leading to these results has received funding from the European Community's Seventh Framework Programme FP7 – 317548

