Generating SPARQL Query Containment Benchmarks using the SQCFramework

Muhammad Saleem[1], Qaiser Mehmood[2], Claus Stadler[1], Jens Lehmann[3,4], and Axel-Cyrille Ngonga Ngomo[1,5]

[1] Universität Leipzig, IFI/AKSW, PO 100920, D-04009 Leipzig
lastname@informatik.uni-leipzig.de
[2] INSIGHT, NUIG, Ireland qaiser.mehmood@insight-centre.org
[3] University of Bonn, Germany jens.lehmann@cs.uni-bonn.de
[4] Fraunhofer IAIS, Bonn, Germany jens.lehmann@iais.fraunhofer.de
[5] University of Paderborn, Germany axel.ngonga@upb.de

Abstract. In this demo paper, we present the interface of the SQCFramework [8], a SPARQL query containment benchmark generation framework. SQCFramework is able to generate customized SPARQL containment benchmarks from real SPARQL query logs. To this end, the framework makes use of different clustering techniques. It is flexible enough to generate benchmarks of varying sizes and complexities according to user-defined criteria on important SPARQL features for query containment benchmarking. We evaluate the usability of the interface by using the standard system usability scale questionnaire. Our *overall usability score of 82.33* suggests that the online interface is consistent, easy to use, and the various functions of the system are well integrated.

1 Introduction

Query containment is the problem of deciding if the result set of a query $Q1$ is included in the result set of another query $Q2$. For example, the result set of a query $Q1$:*"find all restaurants in Berlin"* is included in the result set of the query $Q2$: *"find all restaurants in Germany"*. $Q2$ is called the super-query while $Q1$ is a sub-query of $Q2$. Query containment is used to devise efficient query planners, caching mechanisms, data integration, and view maintenance solutions [2]. For example, in our aforementioned example, the result of $Q1$ can be obtained efficiently from the result of $Q2$.

In recent years, a considerable amount of work on SPARQL query containment has been carried out [3,4,9]. To the best of our knowledge, the SPARQL Query Containment Benchmark (SQC-Bench) [1] is the only benchmark designed to test SPARQL query containment solvers. This benchmark contains a fixed number of 76 query containment tests handcrafted by the authors. While this benchmark contains a variety of tests with varying complexities, the number of tests is fixed and all tests are synthetic. In addition, this benchmark does not allow users to generate benchmarks tailored towards a specific use-case.

To fill this gap, we propose the SQCFramework, a framework for the automatic generation of SPARQL query containment benchmarks from real SPARQL query logs. The framework is able to generate benchmarks customized by its user in terms of various

SPARQL query features (defined in Section 2.1). The framework generates the desired benchmark from query logs by using different clustering methods, while considering the customized selection criteria specified by the user.

2 SQCFramework Benchmark Generation

In this section, we briefly present the benchmark generation process in the SQCFramework[6].

2.1 Input Queries and Important SPARQL Features

Our framework takes a set of queries as input. In this work, we aim to generate benchmarks from real user queries. To this end, we use the Linked SPARQL Queries (LSQ) datasets [6], which provide real queries extracted from the logs of public SPARQL endpoints. A query containment benchmark should comprise queries/tests of varying complexities. Hence, we consider the following query features while generating containment benchmarks: (1) number of entailments/sub-queries, (2) number of projection variables, (3) number of BGPs, (4) number of triple patterns, (5,6) max. and min. BGP triples, (7) number of join vertices, (8) mean join vertex degree, (9) number of LSQ features additional features. Our framework allows generating customized benchmarks based on these features.

2.2 Benchmark Generation

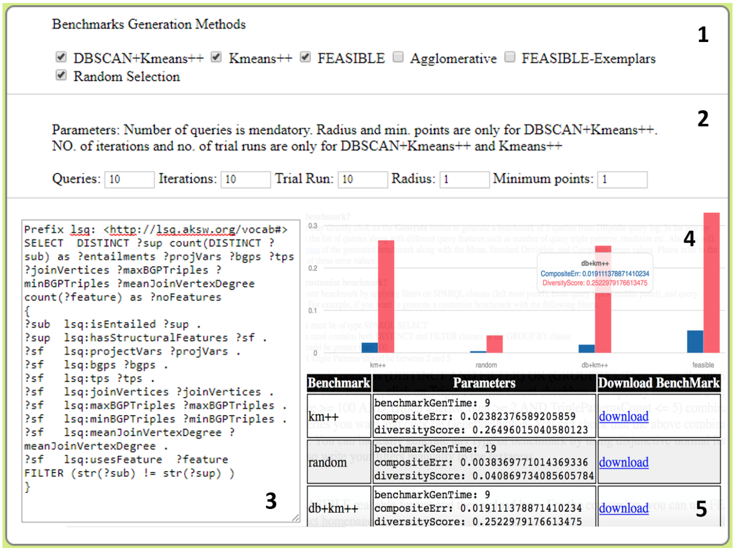
The user provides the input LSQ dataset and the required number N of super-queries to be included in the generated benchmark. Then, the benchmark generation is carried out the following four main steps: (1) Select all the super-queries along with the required features from the input LSQ dataset, (2) Generate feature vectors and their normalization for the selected super-queries, (3) Generate N number of clusters from the super-queries, (4) Select the single most representative super-query from each cluster to be included in the final benchmark.

3 SQCFramework Online

The online demo and source code of the SQCFramework are available at the SQCFramework homepage `https://github.com/AKSW/SQCFramework`. Figure 1 shows the online interface of the SQCFramework, which comprises five main steps:

1. **Selection of benchmark generation method**: The first step is to select the benchmark generation method(s). Currently, our framework supports 6 well-known clustering methods namely DBSCAN+Kmeans++, Kmean++, Agglomerative, Random selection, FEASIBLE [7] and FEASIBLE-Exemplars.

[6] Readers are encouraged to read [8] for complete details and intuition about the selection of a particular choice.


1
2
3
4
5
0.3
0.2
0.1
0
km++
random
db+km++
feasible
db+km++
CompositeErr: 0.019111378871410234
DiversityScore: 0.2522979176613475

Benchmarks Generation Methods

[✓ DBSCAN+Kmeans++] [✓ Kmeans++] [✓ FEASIBLE] Agglomerative FEASIBLE-Exemplars
[✓ Random Selection]

Parameters: Number of queries is mendatory. Radius and min. points are only for DBSCAN+Kmeans++.
NO. of iterations and no. of trial runs are only for DBSCAN+Kmeans++ and Kmeans++

Queries: 10 Iterations: 10 Trial Run: 10 Radius: 1 Minimum points: 1

```
Prefix lsq: <http://lsq.aksw.org/vocab#>
SELECT  DISTINCT ?sup count(DISTINCT ?
sub) as ?entailments ?projVars ?bgps ?tps
?joinVertices ?maxBGPTriples ?
minBGPTriples ?meanJoinVertexDegree
count(?feature) as ?noFeatures
{
?sub  lsq:isEntailed ?sup .
?sup  lsq:hasStructuralFeatures ?sf .
?sf   lsq:projectVars ?projVars .
?sf   lsq:bgps ?bgps .
?sf   lsq:tps ?tps .
?sf   lsq:joinVertices ?joinVertices .
?sf   lsq:maxBGPTriples ?maxBGPTriples .
?sf   lsq:minBGPTriples ?minBGPTriples .
?sf   lsq:meanJoinVertexDegree ?
meanJoinVertexDegree .
?sf   lsq:usesFeature  ?feature
FILTER (str(?sub) != str(?sup) )
}
```

Benchmark	Parameters	Download BenchMark
km++	benchmarkGenTime: 9 compositeErr: 0.02382376589205859 diversityScore: 0.26496015040580123	download
random	benchmarkGenTime: 19 compositeErr: 0.0038369771014369336 diversityScore: 0.040869734085605784	download
db+km++	benchmarkGenTime: 9 compositeErr: 0.019111378871410234 diversityScore: 0.2522979176613475	download

Fig. 1: The SQCFramework online interface

2. **Parameters selection**: The second step is the selection of parameters like the number of queries in the resulting benchmark or the number of iterations for Kmeans++ clustering, etc.
3. **Benchmark personalization**: The third step allows to further customize the resulting benchmark. The SPARQL query selects all the queries along with required features to be considered for benchmarking by default. The user can modify this query to generate customized benchmarks. For example, adding `FILTER(?projVars <=2 && (?bgps > 1 || ?tps >3))` will force the resulting benchmark queries having number of projection variables less than 3 and the number of BGPs greater than 1 or the number of triple patterns greater than 3.
4. **Results**: The diversity score and the similarity errors for the selected methods will be shown as bar graphs.
5. **Benchmarks download**: The resulting benchmarks can be finally downloaded and used in the evaluation of containment solvers.

We will demonstrate all of the steps above to the ISWC audience and generate benchmarks from the LSQ log of the Semantic Web Dog Food dataset.

4 Evaluation

An evaluation of the SQCFramework can be found in [8]. To assess the usability of our system, we used the standardized, ten-item Likert scale-based *System Usability Scale* (SUS) [5] questionnaire[7] which can be used for global assessment of systems usability.

[7] Our survey can found at: `https://goo.gl/forms/n4IG2FQ1PBNCdxBl1`

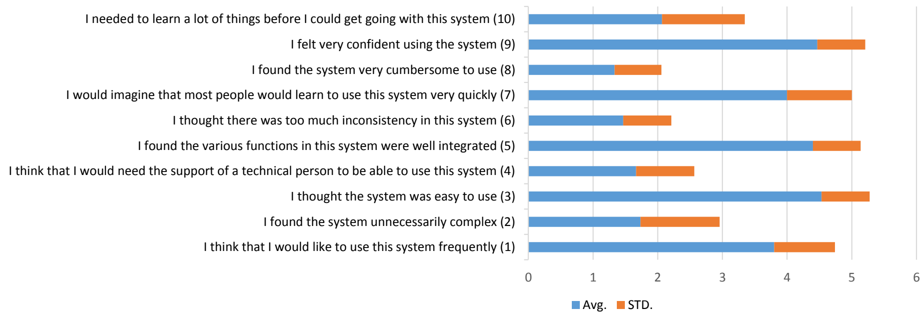


Fig. 2: Result of usability evaluation using SUS questionnaire.

The survey was posted through Twitter with the ISWC2018 hashtag and was filled by 15 users.[8] The results of SUS usability survey is shown in Figure 2. We achieved a mean usability score of **82.33** indicating a high level of usability according to the SUS score. The responses to question 1 suggests that our system is adequate for frequent use (average score to question 1 = 3.8 $\pm$ 0.94) by users all of type. The responses to question 3 (average score 4.53 $\pm$ 0.74) suggests that the interface is easy to use and the responses to question 5 indicates that the various functions are well integrated (average score 4.4 $\pm$ 0.73). However, the response to question 10 (average score 2.06 $\pm$ 1.27) indicates that users need to learn some basic concepts before they can use the system effectively.

Acknowledgment

This work has been supported by the H2020 project HOBBIT (no. 688227).

References

1. M. W. Chekol, J. Euzenat, P. Genevès, and N. Layaïda. Evaluating and benchmarking sparql query containment solvers. In *ISWC*, 2013.
2. C. Chekuri and A. Rajaraman. Conjunctive query containment revisited. *Theoretical Computer Science*, 239(2):211–229, 2000.
3. P. Geneves, N. Layaïda, and A. Schmitt. Efficient static analysis of xml paths and types. *Acm Sigplan Notices*, 42(6):342–351, 2007.
4. A. Letelier, J. Pérez, R. Pichler, and S. Skritek. Static analysis and optimization of semantic web queries. *TODS*, 38(4):25, 2013.
5. J. R. Lewis and J. Sauro. The factor structure of the system usability scale. In *HCD*. 2009.
6. M. Saleem, I. Ali, A. Hogan, Q. Mehmood, and A.-C. Ngonga Ngomo. LSQ: The linked sparql queries dataset. In *ISWC*, 2015.
7. M. Saleem, Q. Mehmood, and A.-C. N. Ngomo. Feasible: a feature-based sparql benchmark generation framework. In *ISWC*, 2015.

[8] As of May 31st, 2018. Summary of the responses can be found at: `https://goo.gl/W5eZkv`

8. M. Saleem, C. Stadler, Q. Mehmood, J. Lehmann, and A.-C. N. Ngomo. Sqcframework: Sparql query containment benchmark generation framework. In *K-Cap*, 2017.
9. C. Stadler, M. Saleem, A.-C. N. Ngomo, and J. Lehmann. Efficiently pinpointing sparql query containments. In *ICWE*, 2018.