Automatic SPARQL Benchmark Generation Using FEASIBLE

Muhammad Saleem[1], Qaiser Mehmood[2], and Axel-Cyrille Ngonga Ngomo[1]

[1] Universität Leipzig, IFI/AKSW, PO 100920, D-04009 Leipzig
{lastname}@informatik.uni-leipzig.de
[2] Insight Center for Data Analytics, National University of Ireland, Galway
qaiser.mehmood@insight-centre.org

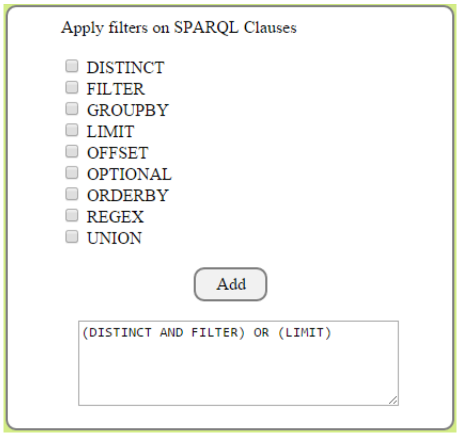
Abstract. Benchmarking is indispensable when aiming to assess technologies with respect to their suitability for given tasks. In this demo, we present the interface of FEASIBLE, an automatic approach for the generation of benchmarks out of the sets of queries. The generation is achieved by selecting prototypical queries of a user-defined size from an input set of queries. In our demo, we focus on the functionality offered by the interface, especially for pre-selecting the queries to be considered while generating the benchmark. We evaluate the usability of the interface by using the standard system usability scale questionnaire. Our overall usability score of 74.06 suggests that FEASIBLE's interface is easy to use, consistent, and the various functions in the system are well integrated.

1 Introduction

Most Linked Data applications rely on triple stores for data storage [6]. The performance of triple stores is hence of central importance for Linked-Data-based software. Several benchmarks have been proposed to assess the performance of triple stores [9]. However, many of these benchmarks rely either on synthetic data or synthetic queries. Previous works [3,2,7,9] point out that artificial benchmarks are mostly unable to reflect the characteristics of real datasets and queries (i.e., query logs). The DBpedia SPARQL Benchmark (DBPSB) [6] addresses a portion of these drawbacks partly by evaluating the performance of triple stores based on real DBpedia query logs. The main drawback of this benchmark is still that it does not consider important query features (e.g., number of join vertices, triple patterns selectivities or query execution times etc.) which greatly affect the performance of triple stores [1,4] during the query selection process. Furthermore, it only considers one of the four SPARQL query forms (i.e., `SELECT`).

These problems are addressed by FEASIBLE [9],[3] a benchmark generation framework able to generate benchmarks from a set of queries (in particular from query logs). FEASIBLE aims to generate customized benchmarks for given use cases or needs of an application. To this end, FEASIBLE assumes that it is given a set of queries as well as the number of queries (e.g., 25) to be included into the benchmark as input. With the FEASIBLE interface, which is the focus of this paper, users are then enabled to choose the queries that they deem relevant for the benchmark generation through

[3] Accepted in the ISWC 2015 research track



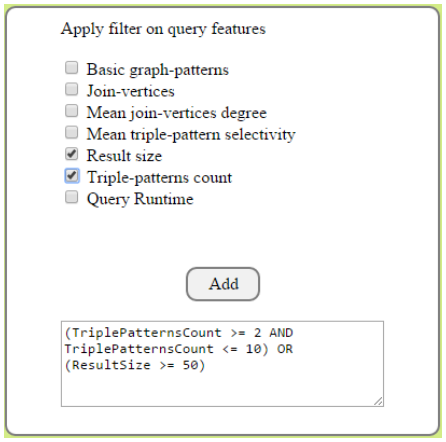
Apply filters on SPARQL Clauses

DISTINCT
FILTER
GROUPBY
LIMIT
OFFSET
OPTIONAL
ORDERBY
REGEX
UNION

Add

```
(DISTINCT AND FILTER) OR (LIMIT)
```

(a) Clauses



Apply filter on query features

Basic graph-patterns
Join-vertices
Mean join-vertices degree
Mean triple-pattern selectivity
[✓ Result size]
[✓ Triple-patterns count]
Query Runtime

Add

```
(TriplePatternsCount >= 2 AND
TriplePatternsCount <= 10) OR
(ResultSize >= 50)
```

(b) Features

Fig. 1: The FEASIBLE filters panels

a series of filters. For example, the users can choose to only include queries with a result size greater than 50 and between 2 and 5 triple patterns. A simple click then launches FEASIBLE on the selected subset of queries and returns a benchmark tailored towards the user's need and requirements. FEASIBLE is open-source and available online at `https://code.google.com/p/feasible/`. A demo can be found at `http://feasible.aksw.org/`. In the following, we present and evaluate the features of FEASIBLE's demo interface.

2 The FEASIBLE Interface

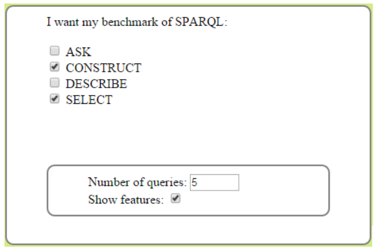
In the demo, we will explain and present the features of the FEASIBLE interface. The main interface comprises three main panels: the clause filter panel, the feature selection panel, and (3) the form selection panel. In the following, we explain each of these panels in details.

2.1 Clause Filter Panel

This panel is used for selecting queries based on the the most commonly used SPARQL clauses [8]. To this end, the user is enable to write a disjunctive normal form of clauses. Only queries which abide by these filters are given the FEASIBLE as input for the benchmark generation. An example filter (`(DISTINCT AND FILTER) OR (LIMIT)`) is shown in Figure 1a. By applying this filter, all of the benchmark queries will either contain both `DISTINCT` and `FILTER` clauses or the `LIMIT` clause.

2.2 Feature Selection Panel

This panel is used for applying filters on those query features which have been shown to greatly affect the performance of triple stores [1,4]. Like in the previous panel, the



I want my benchmark of SPARQL:

- [ ] ASK
- [✓] CONSTRUCT
- [ ] DESCRIBE
- [✓] SELECT

Number of queries: 5
Show features: [✓]

(a) Query forms selection panel

(b) Voronoi diagram of the benchmark

Fig. 2: The FEASIBLE query forms selection panel and the output Voronoi diagram

user can create a disjunction of any conjunction of the query features. In addition, the user can specify the minimum and maximum ranges on the selected feature. An example filter (`(TriplePatternsCount >= 2 AND TriplePatternsCount <= 10) OR (ResultSize >= 50`) is shown in Figure 1a. By applying this filter, all of the benchmark queries will either contain between 2 and 10 triple patterns or the size of their result set will be at least 50.

2.3 Form Selection Panel

This panel is used to select the basic query forms to use in the benchmark, i.e., `SELECT`, `CONSTRUCT`, `DESCRIBE`, and `ASK`. The number of queries to be included in the benchmark can also be selected. The form selection panel shown in Figure 2a will generate a 5-query benchmark with no SPARQL `CONSTRUCT` and `DESCRIBE` queries.

The Voronoi diagram shown in Figure 2b shows the generated 125-queries benchmark along with benchmark queries (highlighted in red) for the DBpedia query log. During the demo, we will present different configurations as well as allow participants to create their own benchmarks. Additional features of our interface include uploading a set of queries as well as downloading the resulting benchmark query by query or as one file.

3 Evaluation

An evaluation of FEASIBLE can be found in [9]. To assess the usability of our system, we used the standardized, ten-item Likert scale-based *System Usability Scale* (SUS) [5] questionnaire[4]. The SUS is a reliable, low-cost usability scale that can be used for global assessments of systems usability[5]. The survey was posted through Twitter with the ISWC2015 hashtag and was filled by 16 users[5] (by 30th June 2015). The results of SUS usability survey is shown in Figure 3. We achieved a mean usability score of **74.06** 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 =

[4] Our survey can found at: `http://goo.gl/forms/UEK4ZQSuYC`

[5] Summary of the responses can be found at: `https://goo.gl/3h1Lkp`

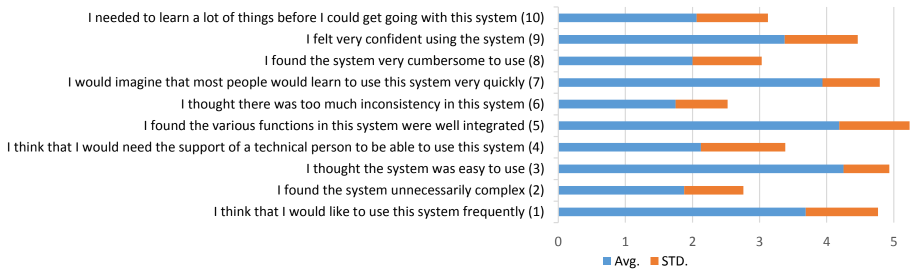


Fig. 3: Result of usability evaluation using SUS questionnaire.

3.67 $\pm$ 1.07) by users all of type. The responses to question 3 (average score 4.25 $\pm$ 0.68) suggests that FEASIBLE is easy to use and the responses to question 5 indicates that the various functions are well integrated (average score 4.18 $\pm$ 1.04).

4 Conclusion and Future Work

In this paper we presented the FEASIBLE demo interface. Our SUS usability score suggest that the majority of the users felt confident using our demo interface. As underlying resource for our future work, we have converted linked data query logs into RDF and made available through LSQ [8] endpoint[6]. Beside the key characteristics discussed in Figure 1, we have attached many of the SPARQL 1.1 features to each of the queries. We will extend FEASIBLE to query this SPARQL endpoint directly to gather queries for the benchmark creation process.

References

1. Güneş Aluç, Olaf Hartig, M Tamer Özsu, and Khuzaima Daudjee. Diversified stress testing of rdf data management systems. In *ISWC*. 2014.
2. Mario Arias, Javier D. Fernández, Miguel A. Martínez-Prieto, and Pablo de la Fuente. An empirical study of real-world SPARQL queries. *CoRR*, 2011.
3. Songyun Duan, Anastasios Kementsietsidis, Kavitha Srinivas, and Octavian Udrea. Apples and oranges: A comparison of rdf benchmarks and real rdf datasets. In *SIGMOD*, 2011.
4. Olaf Görlitz, Matthias Thimm, and Steffen Staab. Splodge: Systematic generation of sparql benchmark queries for linked open data. In *ISWC*. 2012.
5. James R Lewis and Jeff Sauro. The factor structure of the system usability scale. In *HCD*. 2009.
6. Mohamed Morsey, Jens Lehmann, Sören Auer, and Axel-Cyrille Ngonga Ngomo. Dbpedia sparql benchmark - performance assessment with real queries on real data. In *ISWC*, 2011.
7. Francois Picalausa and Stijn Vansummeren. What are real sparql queries like? In *SWIM*, 2011.
8. Muhammad Saleem, Intizar Ali, Aidan Hogan, Qaiser Mehmood, and Axel-Cyrille Ngonga Ngomo. LSQ: The linked sparql queries dataset. In *ISWC*, 2015.
9. Muhammad Saleem, Qaiser Mehmood, and Axel-Cyrille Ngonga Ngomo. FEASIBLE: A featured-based sparql benchmark generation framework. In *ISWC*, 2015.

[6] LSQ homepage: `http://aksw.github.io/LSQ/`