Towards a Top-K SPARQL Query Benchmark

Shima Zahmatkesh, Emanuele Della Valle, Daniele Dell'Aglio, and Alessandro Bozzon

DEIB - Politecnico of Milano, Delft University of Technology
shima.zahmatkesh@polimi.it, emanuele.dellavalle@polimi.it,
daniele.dellaglio@polimi.it, a.bozzon@tudelft.nl

Abstract. The research on optimization of top-k SPARQL query would largely benefit from the establishment of a benchmark that allows comparing different approaches. For such a benchmark to be meaningful, at least two requirements should hold: 1) the benchmark should resemble reality as much as possible, and 2) it should stress the features of the top-k SPARQL queries both from a syntactic and performance perspective. In this paper we propose Top-k DBPSB: an extension of the DBpedia SPARQL benchmark (DBPSB), a benchmark known to resemble reality, with the capabilities required to compare SPARQL engines on top-k queries.

Keywords: Top-k Query, SPARQL, Benchmark

1 Problem Statement

Top-k queries – queries returning the top k results ordered according to a user-defined scoring function – are gaining attention in the Database [1] and Semantic Web communities [2–6]. Order is an important property that can be exploited to speed up query processing, but state-of-the-art SPARQL engines such as Virtuoso [7], Sesame [8], and Jena [9], do no exploit order for query optimisation purposes. Top-k SPARQL queries are managed with a *materialize-then-sort* processing schema [1] that computes all the matching solutions (e.g., thousands) even if only a limited number k (e.g., ten) are requested.

Recent works [2–5] have shown that an efficient *split-and-interleave* processing schema [1] could be adopted to improve the performance of top-k SPARQL queries. To the best of our knowledge, a consistent comparison of those works does not exist. As often occurs, the main cause for this fragmentation resides in the partial lack of a SPARQL benchmark covering top-k SPARQL queries. To foster the work on top-k query processing within the Semantic Web community, we believe that it is the right time to define a top-k SPARQL benchmark.

Following well known principles of benchmarking [10], we formulate the research question of this work as: *is it possible to set up a benchmark for top-k SPARQL queries, which resembles reality as much as possible and stresses the features of top-k queries both from a syntactic (i.e., queries should contain rank-related clauses) and performance (i.e., the query mix should insist on characteristics of top-k queries which stress SPARQL engine) perspective?*

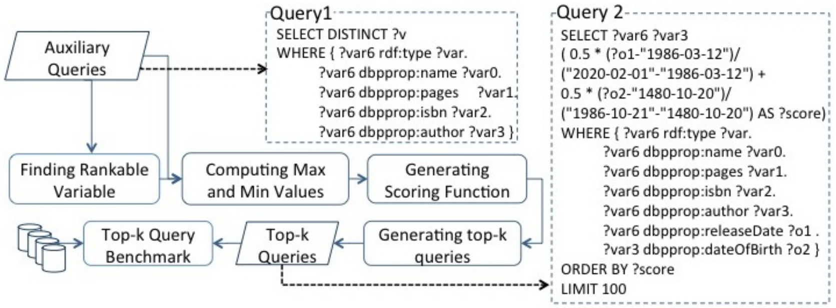
2 Metodology

In this poster, we describe our ongoing work on Top-k DBpedia SPARQL Benchmark (Top-k DBPSB). It extends the methodology, proposed in DBPSB [11], to build SPARQL benchmarks that resemble reality. It uses the same dataset, performance metrics, and test driver of DBPSB, but it extends the *query variability* feature of DBPSB by proposing an algorithm to automatically create top-k queries from the 25 auxiliary queries of the DBPSB and its datasets.

In order to present Top-k DBPSB, we need to introduce some terminology:

- **Definition 1**: *Rankable Data Property* is a RDF property whose range is in xsd:int, xsd:long, xsd:float, xsd:integer, xsd:decimal, xsd:double, xsd:date, xsd:dateTime, xsd:time, or xsd:duration.
- **Definition 2**: *Rankable Triple Pattern* is a triple pattern that has a Rankable Data Property in the property position of the pattern.
- **Definition 3**: When a variable, in the object position of a Rankable Triple Pattern, appears in a scoring criteria of the scoring function, we call it *Scoring Variable* and we call *Rankable Variable* the one appearing in the subject position.

Figure 1 shows an overview of the process followed by Top-k DBPSB to generate top-k SPARQL queries from the Auxiliary Queries and datasets of DBSBM. The process consists of four steps: 1) finding rankable variables, 2) computing maximum and minimum values for each rankable variable, 3) generating scoring functions, and 4) generating top-k queries.



```
SELECT DISTINCT ?v
WHERE { ?var6 rdf:type ?var.
        ?var6 dbpprop:name ?var0.
        ?var6 dbpprop:pages    ?var1.
        ?var6 dbpprop:isbn ?var2.
        ?var6 dbpprop:author ?var3 }
```

```
SELECT ?var6 ?var3
( 0.5 * (?o1-"1986-03-12")/
("2020-02-01"-"1986-03-12") +
0.5 * (?o2-"1480-10-20")/
("1986-10-21"-"1480-10-20") AS ?score)
WHERE { ?var6 rdf:type ?var.
        ?var6 dbpprop:name ?var0.
        ?var6 dbpprop:pages ?var1.
        ?var6 dbpprop:isbn ?var2.
        ?var6 dbpprop:author ?var3.
        ?var6 dbpprop:releaseDate ?o1 .
        ?var3 dbpprop:dateOfBirth ?o2 }
ORDER BY ?score
LIMIT 100
```

Fig. 1: Top-k DBPSB generates top-k SPARQL queries starting from the Auxiliary queries and datasets of DBPSB (a SPARQL benchmark known to resemble reality).

In the first step, Top-k DBPSB looks for all rankable variables, in each auxiliary query of each BDPSB query template, which could be used as part of a ranking criterion in a scoring function. For each DBPSB auxiliary query, Top-k DBPSB first checks if the variables in the query fit the definition of scoring variable. To find additional rankable variable Top-k DBPSB considers all variables in the query and tries also to find rankable triple pattern related to them.

For the sake of simplicity, Top-k DBPSB generates scoring function as a weighted sum of normalized ranking criteria, therefore, for each rankable variable, Top-k DBPSB needs to compute its maximum and minimum value. So, for each DBPSB auxiliary query and each rankable triple pattern identified in the previous step, Top-k DBPSB generates a query to find those values.

After having collected those values, for each rankable triple pattern, Top-k DBPSB creates a new top-k query template. For instance, Query 2 of Figure 1 shows such a query as generated by the Top-K DBPSB. In order to obtain an executable query, for each scoring variable that appears in scoring function Top-K DBPSB adds the related rankable triple pattern to the body of the query. [1].

3 Experiments

Given that we extended DBPSB we give for positively answered the first part of our research question (i.e., Top-k DBPSB resembles reality). In this section, we provide preliminary evidence that the query variability feature of Top-k DBPSB positively answers also the second part of our research question. We do so by operationalising our research question in three hypotheses:

H.1 The more are the number of the Rankable Variables, the longer is the average execution time.

H.2 The more are the number of the Scoring Variables in the scoring function, the longer is the average execution time.

H.3 The value of the `LIMIT` clause has not any significant impact on the average execution time.

In order to evaluate our hypothesis, we carry out our experiments by using the SPARQL engines Virtuoso, and Jena. After preparing the experimental environment, we load the DBpedia dataset with the scale factors of 10% on the SPARQL engines. In order to evaluate the performance of SPARQL engines, we use the DBpedia SPARQL Benchmark test driver and modify it for Top-k queries. We execute the generated Top-k SPARQL queries against these two SPARQL engines. The information gains from the execution of randomly generated query against the SPARQL engines is used to evaluate our hypotheses.

As a result we got that most of the queries templates generated by Top-k DBPSB are compatible with our hypotheses H.2 and H.3. On the contrary, Top-k DBPSB does not generate adequate query templates to test the hypothesis H.1: only 6 queries templates out of the 25 in DBPSB can be use in validating H.1 while the others have only one rankable variable. Further investigation is needed to refine the hypothesis H.1 and better qualify the cases that stress SPARQL engines when answering top-k queries.

[1] The implementation code is available online at $https : //bitbucket.org/sh\_zahmatkesh/top - k - dbpsb$

4 Conclusion

In this work, we presented Top-k DBPSB, an extension of DBPSB [11] that adds the possibility to automatically generate top-k queries. Top-k DBPSB satisfies the requirement of resembling the reality extending DBPSB which automatically derives datasets and queries from DBpedia and it query logs. Moreover, we provide experimental evidence that Top-k DBPSB also satisfies the requirement to stress the distinguish features of top-k queries.

In order to support the claim that we positively answered to the research question presented in Section 1, we experimentally showed that the query variability provided by Top-k DBPSB stresses the SPARQL engines. To this end, we formulated three hypothesis and we empirically demonstrated that when the number of scoring variables increases the average execution time also does (hypothesis H.2) and that the average execution time is independent from the value used in the `LIMIT` clause (hypothesis H.3). Counterintuitively, hypothesis H.1 is not confirmed by our experiments.

References

1. Ilyas, I.F., Beskales, G., Soliman, M.A.: A survey of top-k query processing techniques in relational database systems. ACM Computing Surveys (CSUR) **40**(4) (2008) 11
2. Magliacane, S., Bozzon, A., Della Valle, E.: Efficient execution of top-k sparql queries. In: The Semantic Web–ISWC 2012. Springer (2012) 344–360
3. Wagner, A., Duc, T.T., Ladwig, G., Harth, A., Studer, R.: Top-k linked data query processing. In: The Semantic Web: Research and Applications. Springer (2012) 56–71
4. Cheng, J., Ma, Z., Yan, L.: f-sparql: a flexible extension of sparql. In: Database and Expert Systems Applications, Springer (2010) 487–494
5. Cedeno, J.P.: A Framework for Top-K Queries over Weighted RDF Graphs. PhD thesis, Arizona State University (2010)
6. Siberski, W., Pan, J.Z., Thaden, U.: Querying the semantic web with preferences. In: ISWC. (2006) 612–624
7. Erling, O., Mikhailov, I.: Rdf support in the virtuoso dbms. In Auer, S., Bizer, C., Müller, C., Zhdanova, A.V., eds.: CSSW. Volume 113 of LNI., GI (2007) 59–68
8. Broekstra, J., Kampman, A., van Harmelen, F.: Sesame: A generic architecture for storing and querying rdf and rdf schema. In Horrocks, I., Hendler, J.A., eds.: International Semantic Web Conference. Volume 2342 of Lecture Notes in Computer Science., Springer (2002) 54–68
9. Owens, A., Seaborne, A., Gibbins, N., mc schraefel: Clustered tdb: A clustered triple store for jena. Technical report, Electronics and Computer Science, University of Southampton (2008)
10. Gray, J., ed.: The Benchmark Handbook for Database and Transaction Systems (2nd Edition). Morgan Kaufmann (1993)
11. Morsey, M., Lehmann, J., Auer, S., Ngomo, A.C.N.: Dbpedia sparql benchmark - performance assessment with real queries on real data. In Aroyo, L., Welty, C., Alani, H., Taylor, J., Bernstein, A., Kagal, L., Noy, N.F., Blomqvist, E., eds.: International Semantic Web Conference (1). Volume 7031 of Lecture Notes in Computer Science., Springer (2011) 454–469