QUARQ: QUick Approximate and Relaxed Querying

Victor-Alejandro Ortiz*†, Maria-Cristina Marinescu*, Maria-Ribera Sancho*†
*Barcelona Supercomputing Center (BSC), Barcelona, Spain
†Universitat Politècnica de Catalunya (UPC), Barcelona, Spain
E-mail: {victor.ortiz, maria.marinescu, maria.ribera}@bsc.es

***Keywords*—Ontologies, Semantic Web, Linked Data, query performance, SPARQL, Online Learning.**

I. Extended Abstract

Executing queries over Linked Open Data (LOD) is a complex task. The total number of sources triggered by a single query cannot be known in advance, nor the reasoning complexity applied to each source. In order to avoid this uncertainty, practitioners download full replicas of the open data and build applications on top of the datasets in a controlled environment. With this centralized approach, they lose dynamic data changes, and often they cannot account for the inference capabilities defined in the associated ontologies.

In this work, we explore the feasibility of predicting the performance of Flexible Querying over Linked Open Data [1]. Concretely, we propose QUARQ: QUick Approximate and Relaxed Querying, a tool that using ML provides intelligence to the process of generating alternative queries that run more efficiently than the original ones. With this tool, we propose avoiding the use of replicated Linked Data by seizing the shareable nature of Linked Data and eluding the impracticality of maintaining copies up-to-date or the need to work with outdated data.

A. Overview of the solution

Our solution designs and implements a tool aimed at the Semantic Web practitioners, whose purpose is to query LOD directly through the available public SPARQL endpoints. QUARQ uses ML under an Online Learning (OL) paradigm to predict the performance of a query and, based on parameters provided by the user, generate alternate queries that run more efficiently at the cost of accuracy.

We take the LOD-Cloud diagram as a reference for our project, the most adopted graph of links between multiple datasets. The LOD-cloud provide us with datasets statistics on their published resources. Moreover, a larger part of the Semantic Web community uses this graph to query LOD resources and make available their datasets to the community following Linked Data principles. The diagram currently contains 1301 datasets with 16283 links [1] although not all of them are continuously available [2].

We propose that selecting the flexible querying method with the best performance at a given time to transform an original user query is a task that ML techniques can learn by using OL to predict the performance of alternative queries generated by different Approximate and Relaxed Querying algorithms [3].

[1]https://lod-cloud.net/ consulted on March 28th, 2022

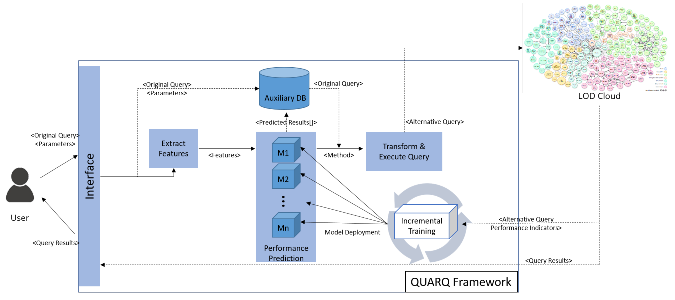


Fig. 1. QUARQ architechture

QUARQ selects the best possible method to generate an alternate query and executes it directly in public endpoints provided by the datasets published as part of the LOD-cloud.

a) Query Approximation: An Approximate Query (AQ) simplifies the execution process of the query by using data sampling or performing a data synopsis to shorten the response time at the cost of accuracy; it does not rewrite the query [4].

b) Query Relaxation: A Relaxed Query (RQ) rewrites the query to use alternative concepts when the original one cannot return data, results in a query that does not terminate, or can only do so in an amount of time that is not reasonable, ranking the results of a query depending on how closely the original predicate of the query is satisfied [4].

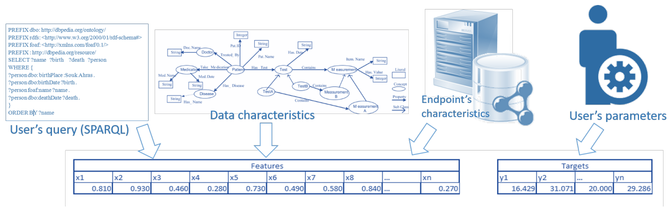
B. Architecture of the solution

Our goal is to build a tool that provides the final user with easy and scalable access to a broad amount of interconnected data, regardless of the user's prior knowledge of data sources, schemas, or technical skills necessary to formulate efficient queries. QUARQ acts as an intermediary between the user and LOD-cloud - through public endpoints, guaranteeing the user will get an approximate valid result and avoiding empty responses.

Figure 1 illustrates QUARQ's architecture, which comprises the following components:

1) Interface: The user interface allows users to input the query they want to execute over the LOD cloud. Moreover, it also provides a parameters settings functionality, where the user inputs levels of runtime, precision and size of the retrieved data from the query. Finally, a query results visualization.

Input and visualization of results are traditional functions on a query interface. Nonetheless, the critical component in



```
PREFIX dbo: http://dbpedia.org/ontology/
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX : http://dbpedia.org/resource/
SELECT ?name  ?birth  ?death  ?person
WHERE {
?person dbo: birthPlace :Souk Ahras .
?person dbo:birthDate ?birth .
?person foaf:name ?name .
?person dbo:deathDate ?death .
}
ORDER BY ?name
```

Features									
x1	x2	x3	x4	x5	x6	x7	x8	...	xn
0.810	0.930	0.460	0.280	0.730	0.490	0.580	0.840	...	0.270

Targets			
y1	y2	...	yn
16.429	31.071	20.000	29.286

Fig. 2. Feature extraction representation

this section is the parametrization, which is indicative of the user's expectations and used as a target for ML models to predict the queries performance under different alternate query scenarios. The user defines three parameters simultaneously and prioritizes the importance of each one. We can assign a weighted value to each parameter and define an individual assessment with a combined result on the multiple parameter prediction.

2) Feature extraction: Our tool makes predictions based on features derived from an initial input provided by the user - query, endpoint URL and parameters; however, the feature selection process should be thoroughly executed to avoid saturating our model with irrelevant features.

As illustrated in Figure 2, our solution, takes the input provided by the user and extracts different features derived from:

- Original query provided by the user
- Referenced datasets
- Characteristics of endpoints
- Parameters

Based on a previous work presented by [5], we hypothesize that the relevant features are contained in these main sources. However, the objective of this section is to identify and extract from the query only those characteristics that add value to our model, disregarding those irrelevant to the performance's prediction to avoid poor quality or unreliable predictions.

3) Performance prediction: Our solution applies online learning [6], an ML paradigm that does not require all the data upfront. This approach is concerned with improving the models using new incoming data. OL-based models remember previous data as a dynamic training vector without reprocessing it.

The performance prediction component receives the set of features extracted from the original query and runs a regression analysis to predict the expected parameters' values for each of the flexible querying methods implemented in QUARQ.

4) Query Transformation and execution: The query transformation component is in charge of rewriting the original query -given by the user - by applying the flexible querying algorithm determined that is most likely to obtain the targeted performance. This querying method returns either approximate results by applying a Query Approximation (QA) method or relaxes the conditions of a query to avoid empty responses using a Query Relaxation method (QR).

5) Incremental Training: In principle, Incremental learning models - also known as Online learning or data stream learning [7] - allow for both data prediction and retraining requests. The incremental training component is the most critical section of the tool and gives the most added value. Its contribution to our solution is twofold: it allows our predictor to learn in an *empirical* way while granting a level of adaptability to a continuously changing environment.

Every time a user formulates and executes a query, our tool returns results and values for metrics on runtime, accuracy and number of responses. These metrics are labelled as targets to retrain the models that predict the performance for each of the implemented Flexible Querying methods. To avoid delays in the execution of prediction tasks, we propose to perform the retraining process separately on an offline copy. The old model is substituted by a new retrained version of the model, and another new copy stays offline to start the retraining process again.

C. Conclusion

Our tool allows Linked Data practitioners to query the so-called "web of data" in a more efficient way by means of sacrificing some precision but under parameters defined by the user themselves. This enables any user to avoid blank results and always rely on having an answer that approaches their needs regarding runtime, completion or precision.

References

[1] R. Hasan, "Predicting SPARQL Query Performance and Explaining Linked Data," pp. 795–805, 2014.

[2] J. A. Nasir and J. P. McCrae, "iLOD: Interplanetary file system based linked open data cloud," *CEUR Workshop Proceedings*, vol. 2821, no. 825182, pp. 27–32, 2020.

[3] R. Frosini *et al.*, "Flexible query processing for SPARQL," *Semantic Web*, vol. 8, no. 4, pp. 533–563, 2017.

[4] S. Chaudhuri *et al.*, "Approximate Query Processing: No silver bullet," *Proceedings of the ACM SIGMOD International Conference on Management of Data*, vol. Part F1277, pp. 511–519, 2017.

[5] R. Hasan and F. Gandon, "A machine learning approach to SPARQL query performance prediction," *Proceedings - 2014 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology - Workshops, WI-IAT 2014*, vol. 1, pp. 266–273, 2014.

[6] J. Liu and E. Zio, "An adaptive online learning approach for Support Vector Regression: Online-SVR-FID," *Mechanical Systems and Signal Processing*, vol. 76-77, pp. 796–809, 2016. [Online]. Available: http://dx.doi.org/10.1016/j.ymssp.2016.02.056

[7] H. M. Gomes *et al.*, "Machine Learning for Streaming Data: State of the Art, Challenges, and Opportunities," *SIGKDD Explor. Newsl.*, vol. 21, no. 2, pp. 6–22, 2019. [Online]. Available: https://doi.org/10.1145/3373464.3373470

Victor-Alejandro Ortiz received his BSc. degree in Informatics from the National Autonomous University of Mexico (UNAM), and the MSc. degree in Big Data and HPC from the University of Liverpool, UK. Currently pursuing a Ph.D. in Computer Science at the Universitat Politècnica de Catalunya (UPC) and the Barcelona Supercomputing Center (BSC), Spain.