RDF-ANALYTICS: Interactive Analytics over RDF Knowledge Graphs

Maria-Evangelia Papadaki and Yannis Tzitzikas
Institute of Computer Science, FORTH-ICS, 70013 Heraklion, Greece, and
Computer Science Department, University of Crete, Greece
marp@ics.forth.gr, tzitzik@ics.forth.gr

ABSTRACT
The formulation of structured queries in knowledge graphs is a
challenging task that presupposes familiarity with the syntax of
the query language and the contents of the knowledge graph. To
alleviate this difficulty in this paper we introduce RDF-ANALYTICS,
a novel system that enables plain users to formulate analytic
queries over complex, i.e. not necessarily star-schema based, RDF
knowledge graphs. To come up with an intuitive interface, we
leverage the familiarity of users with Faceted Search (FS) systems,
i.e. we extend the familiar interface with actions that enable users to formulate
analytic queries, too. Distinctive characteristics of the approach
is the ability to include arbitrarily long paths in the analytic query
(accompanied with count information), interactive formulation
of HAVING restrictions, the support of both Faceted Search (i.e.
the locating of the desired resources in a faceted search manner)
and analytic queries, and the ability to formulate nested analytic
queries. Finally, we present the results of a preliminary task-based
evaluation with users, which are very promising.

KEYWORDS
Knowledge Graphs, Analytics, Faceted Search

1 CONTEXT AND MOTIVATION
There are several Knowledge Graphs (KGs), i.e. collections of
facts in the form "(subject, relation, object)" expressed in RDF,
that integrate data from various sources: from general purpose,
like DBpedia [3] and Wikidata [30], to domain specific reposito-
ries, e.g., Europeana [13], DrugBank [31], GRSF [28], ORKG [14],
WarSampo [16], [5, 6] for Covid-19, and [7] for digital human-
ities. It would be very useful if plain users could analyze such
interesting but complex amounts of data, interpret it, discover
useful information and derive insights from it in an easy and
flexible way.

2 CHALLENGES
Although, users can browse KGs through the provided web pages
(in case dereferenceable URIs are supported), search them using
keyword search (e.g. through [19]), or query them through plain
SPARQL or through interactive query formulators (like [4] and
[17]), there is not any standard method of formulating analytic
queries. Indeed, the analysis of KGs in RDF is challenging since
knowledge of the terminology and the syntax of query language
are required. Such requirements are quite cumbersome for ordi-
nary users and time-consuming for expert users.

2.1 Running Example
Suppose a KG with information about products and their related
entities, e.g. companies, persons, locations, etc., with schema as
shown in Figure 1 (for reasons of brevity namespaces are hidden).

Figure 1: The schema of the running example

Assume that we would like to answer a query of the form like
"average price of laptops made in 2021 from US companies that have
2 USB ports and an SSD drive manufactured in Asia grouped by
manufacturer". Such a query is quite complex, since it requires
expressing several restrictions that also involve paths in the KG.
Its expression in SPARQL is shown in Fig. 2.

Figure 2: Expression in SPARQL of the query of §2.1.

3 DIRECTION AND APPROACH
To alleviate the aforementioned difficulty, we propose an interac-
tion model that allows plain users to compose analytic queries
through simple clicks (or simple selections), while exploring the
contents of a KG even if they have no technical background. We
aim at finding a generic interaction model that can be applied to
any RDF dataset (not only to datasets that have a star schema) and that guides users to create only answerable queries, "protecting" them from spending effort on trying to formulate queries that are not answerable due to lack of data.

In particular, we leverage the familiarity of users with Faceted Search (FS) [23], since this model lets users express complex conditions through simple clicks. We start from a general model for faceted search over RDF data [27] and we extend it with actions that enable formulating analytic queries. The proposed model supports only answerable queries, restrictions, HAVING clauses, nested queries, paths in both FS and analytic queries, while it provides count information in any state.

Specifically, the classical FS interface usually comprises two main frames: the left one that is used for the facets (filters), and the right one that is used for showing the objects, see Figure 3(left). We propose enriching the user actions of the left frame with actions for formulating analytic queries, specifically notice the □ and G buttons on each facet shown in Figure 3(right): the first for specifying grouping function(s), the second for the measuring function(s). In addition, we need one additional frame, let call it Answer Frame, for short 𝐴𝐹, for showing the results of the analytic query (in a tabular or other method). To enable the formulation of HAVING restrictions we propose a button "Explore with FS" in the Answer Frame, through which the user can load the results of the current query as a new dataset, and can (again through FS) specify restrictions. The latter restrictions correspond to HAVING restrictions over the original dataset.

Figure 3: The core elements of the GUI for Faceted Search and Analytics

4 THE SYSTEM RDF-ANALYTICS

We have implemented the aforementioned ideas as a web application, that we call RDF-ANALYTICS. The server-side uses the triplestore Virtuoso¹ that offers persistent storage and SPARQL endpoint, while the front-end side of the system was based on Angular².

For example, for formulating the query in the running example (§2.1), i.e. "average price of laptops made in 2021 from US companies that have 2 USB ports and an SSD drive manufactured in Asia grouped by manufacturer", the user has to express in a FS manner the condition "laptops made in 2021 from US companies that have 2 USB ports and an SSD drive manufactured in Asia", and uses one button for specifying the "grouped by" and another for specifying the "avg price".

Figure 4 shows a screenshot of RDF-ANALYTICS: On the left of each facet name, there is an expansion icon, i.e. ">>", enabling the user to see the top-level sub-classes and the applicable properties. On the right of each facet name, there is a check-box and three buttons:

- □: for filtering the results (through values of that facet)
- G: for grouping the results (with respect to that facet)
- ±: for selecting the function, i.e. avg, min, max, etc, that will be applied to each group of the analytic query
- >>: for expanding a property path (unlimited depth, bidirectional)

Now Figure 5 shows how the user can restrict the numeric values of a facet within intervals by specifying the minimum and maximum values of them, and how derived attributes e.g. YEAR, MONTH, DAY of a Date, can be extracted. For example, in the running example where the user wants to group the laptops by year, (s)he would click on the grouping button that is next to the facet of the "date" and then (s)he would select the derived attribute of "year" from the provided menu.

Figure 5: Setting intervals

The results of an analytic query are presented in a tabular form and as a plot as shown in Figure 6 for the query of the running example. It is important to stress that if the user clicks on the button "Explore with FS" that is provided below the analytical results, (s)he can also load them, as a new dataset (as shown in the bottom part of Figure 6). Then (s)he can proceed in formulating restrictions and group by queries. This enables the formulation of HAVING restrictions (and nested queries of unlimited depth).

5 EXPRESSIVENESS

Our main objective is to cover common needs in a familiar interaction style, not to propose an interaction model with very high expressive power but too complex to use. Nevertheless, the
resulting model is very expressive enabling the expression of analytic queries that involve complex restrictions that involve property paths. Furthermore, the ability to load the results of a query, as a new dataset over which the user can continue query formulation, enables the formulation of analytic queries with HAVING clause in an intuitive manner.

From the perspective of OLAP operations [29], i.e. roll up (aggregate data by ascending concept hierarchy), drill-down (navigate from less detailed data to more detailed data), slice (perform a selection on one dimension of the given cube), dice (describe a sub-cube by operating a selection on two or more dimensions), pivot (provide an alternative presentation of the data), the interaction of RDF-ANALYTICS supports all of them. In particular, traversing up the hierarchy of a facet corresponds to roll-up, traversing down the hierarchy of a facet corresponds to drill-down, picking one value for a facet corresponds to slice, picking two or more values from multiple facets corresponds to dice, and moving to a facet which is directly or indirectly connected to the facet of focus corresponds to pivot.

6 EFFICIENCY
The information required by the interaction model and the analytics is gained through SPARQL; i.e. the system gradually builds the SPARQL query that will be sent to the SPARQL endpoint. This enables the application of the system to various endpoints. The various techniques that have been proposed in the literature for the optimization of SPARQL analytic queries, e.g. [11, 12] could be investigated for applying them to our system as well; this is a topic that goes beyond the scope of the current demo paper.

7 RELATED WORK AND NOVELTY
In general, there are not so many works, neither running systems (for a recent survey, see [21]).

There are a few works that support the formulation of analytic queries directly over RDF, for instance [9] supports guided query building (including analytic queries) with an implementation over the SPARQLIS editor [8]. As regards expressiveness, HAVING restrictions are not supported, neither count information during query formulation, reducing in this way that exploratory characteristics of the process. In addition, the GUI is not the classical of FS, so it is not familiar to everyone. Another work that falls in this category is [25] that describes a possible extension of SemFacet [15] to support numeric value ranges and aggregation. That paper investigates theoretical query management aspects, it lacks an interface and implementation. Moreover, that model does not support explicit path expansion; instead the authors use the notion of “recursion” to capture reachability-based facet restrictions. Towards the direction of our work (analytics directly over RDF), [20] analyzed the applicability of an abstract language for analytics (HIFUN [26]) over RDF, and provided the algorithms for translating HIFUN queries to SPARQL queries. However, the interactive formulation of a HIFUN query is missing from that work. In the current work we want to fill this gap, since in knowledge graphs with broad coverage it is difficult to find and select the right property let alone the formulation of restrictions.

Another direction is the definition of a data cube over RDF, i.e. there are works that implicitly define a data cube over existing RDF graphs [1, 2, 32], and then apply OLAP. One weakness of this approach (as stressed also in [9]) is that requires someone with technical knowledge to define the required data cube(s). Apart from reduced flexibility, the user cannot leverage the wealth of connections of the knowledge graph, since the user is restricted on the data cube.

Another related topic is the publishing of statistical data, specifically the adoption of the RDF data cube vocabulary for publishing and exchanging statistical data using the W3C RDF standard (e.g. [24]).

There are also, domain specific works (focusing on a particular topic, not on any RDF dataset), like [10] that motivates knowledge graph-enabled cancer data analytics, or [18] that describes an analogous work for covid-19 related data. Such works describe domain-specific pipelines for constructing the desired knowledge graph, for supporting particular analytic queries and visualizations. Such works do not aim at providing general-purpose methods for knowledge graph analytics.

7.1 Our position and contribution
We presented an approach with the following key characteristics: (i) it guides the user in query formulation, and the process never leads to empty results, (ii) it supports both FS and analytic queries, (iii) it supports HAVING clauses, (iv) it supports counts and paths in both FS and analytic queries, and (v) it leverages the familiarity of users with FS.

8 EVALUATION
Comparison with related systems. In Table 1, we compare the two most related systems to our approach, according to some important functionalities, i.e. applicability (application on star schemas or over any RDF graph), support of basic analytic queries, support of analytic queries with HAVING, support of plain Faceted Search, support of property paths in faceted search and analytics, visualization, as well as if there are running systems, and if they have been evaluated. We can notice that RDF-ANALYTICS has the highest number of supported features.

Task-based Evaluation with Users. We performed a small-scale evaluation with users to investigate if they can formulate easily analytic queries (especially queries that include path expressions), and to gain a general feedback from them. We defined 10 tasks and 10 users have participated, so far. We did not train them; we just provided them a concise help page explaining the actions of the buttons. The results so far are very promising in [3].

1https://team.inria.fr/oak/projects/warg/
2https://www.w3.org/TR/vocab-data-cube/
terms of task completion (success 73%, partial success 2%, fail 25%) and user rating (Very useful 50%, Useful 50%, Little Useful 0%, Not Useful 0%). We plan to perform a more precise evaluation (with more users and more tasks) and it will be enriched with more visualization capabilities.

9 CONCLUDING REMARKS

By implementing RDF-Analytics we demonstrated the feasibility of exploring and formulating analytic queries over arbitrary knowledge graphs, without presupposing knowledge neither of the terminology of the data, nor of the query language. Distinctive characteristics of the presented approach is that (i) it guides the user in query formulation, and the formulation process never leads to empty results, (ii) it supports both Faceted Search and analytic queries, (iii) it supports complex restrictions and path expressions, (iv) it supports HAVING restrictions and nested queries, (v) it leverages the familiarity of users with FS, and (vi) it can be applied directly over a SPARQL endpoints. The results of the small-scale evaluation with users provided evidence that the users can use this approach to formulate analytic queries.

The deployment of the system used is accessible through http://62.217.127.128:8080/Interactive-RDF-Analytics/ and is under continuous improvement. In future we plan to investigate the extension of the model with transformation operators that the user would apply at interaction time, for handling empty values and multi-valued properties. In addition, we plan to investigate issues related to optimizations for efficiency.

Acknowledgments

This work has received funding from the European Union’s Horizon 2020 coordination and support action 4CH (Grant agreement No 101004468).

REFERENCES