Christophe Debruyne and Declan O'Sullivan. Visually Exploring SPARQL Endpoints with Murmuration. In EKAW 2020 Posters and Demos.

Visually Exploring SPARQL Endpoints with Murmuration

Christophe Debruyne, Declan O'Sullivan

ADAPT Centre, Trinity College Dublin, Dublin 2, Ireland
{debruync,declan.osullivan}@tcd.ie

Abstract. We present Murmuration, which is a tool to discover various paths between resourced in a Linked Data dataset serviced by a SPARQL endpoint. Murmuration was developed to run entirely on a client. While this computation is heavy on the client, we provide a reasonably smooth experience by generating a series of queries that cover the different possible paths between two resources. This allows us to use, for each query, a callback to display the results as they come along. This approach also reduces the complexity of queries that the SPARQL endpoint has to process. Future work will look into scalable approaches for distributed computing, as we recognize the need for exploring multiple Linked Data datasets as a whole.

Keywords: Linked Data exploration, Linked Data visualization, relationship discovery

1 Introduction

Exploring Linked Data datasets in a visual manner still remains a challenge. A fairly recent overview was provided by [2]. The authors identified and classified tools based on several characteristics, e.g., ontology (schema) exploration vs. RDF exploration, their capabilities, and whether they were a web application. In the context of a project, we needed a tool that allowed one to find arbitrary paths between RDF resources.

Our tool, called Murmuration, is inspired by both RelFinder [3] and graphVizDB [1]. RelFinder iteratively searched for all possible paths between two resources, showing intermediate results when they are found. RelFinder furthermore allowed one to provide the nodes from which the tool has to start from and the possibility filter out certain predicates. The disadvantage of RelFinder is that it relied on now dated technology (Flash) and required setting up a server for some of the PHP scripts. Similarly, graphVizDB requires a server as it stores information about the graphs in a bespoke database. What we appreciate about graphVizDB, however, is that the frontend relies on Web standards. Unlike RelFinder, however, it allows one to search the graphs as a whole and does not look for paths between sources in a graph. Ideally, however, we would have access to a data exploration tool that allows one to:

- look for all paths between resources provided by a user (similar to [3]);
- declare which predicates can be omitted before searching;

- avail of open Web standards for the visualization (similar to [1] and [4]); and
- *solely* rely on client-side processing (similar to [4]).

The second requirement is interesting when users know, beforehand, which predicates appear often and do not provide much additional information such as `rdf:type` and `cidoc:P2_has_type`. Such a feature is not present in the aforementioned tooling; one can filter out predicates when found after a search.

We thus present Murmuration, a tool that allows one to explore all paths between two or more resources. Sections 2 and 3 provide details on the design and implementation, with a focus on the queries that it generates to manage client-side processing. In Section 4, we conclude the paper.

2 Design and Implementation

Murmuration was inspired by RelFinder and graphVizDB, though we wish to develop a tool that relied on client-side processing. An image is provided in Fig. 1. The tool consists of a form where one can configure the search, a canvas on which the nodes and edges are displayed, and a display. Upon clicking on a resource in the visualization, the tool will try to load an HTML representation of that resource in the display.

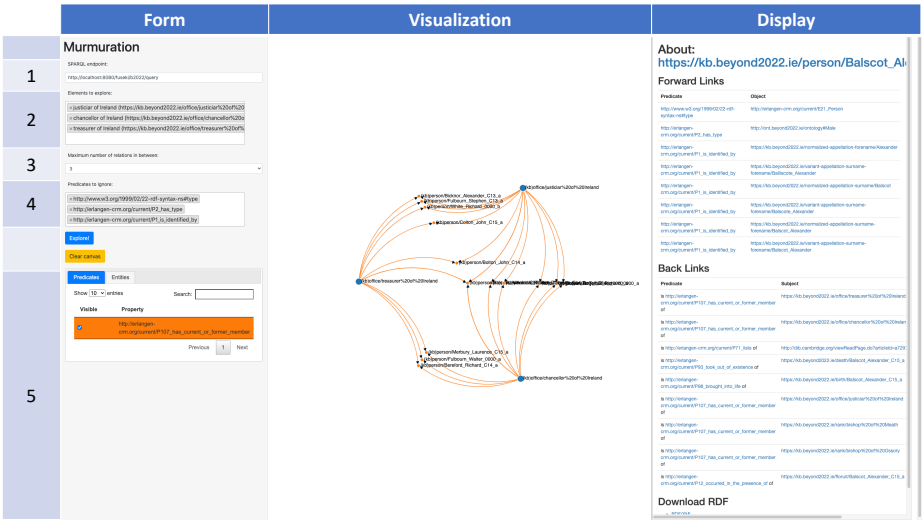

Fig. 1. Demonstrating Murmuration in the Beyond 2022 Project. An endpoint is chosen in (1). We can select resources in (2), which looks for URIs or resources with an `rdfs:label` with particular string for which we want to find paths. (3) allows us to set the maximum of intermediate concepts. (4) allows us to exclude predicates appearing in the dataset. Finally, (5) allows us to toggle nodes and edges in the visualization.

Fig. 1 demonstrates how one can easily discover people that shared multiple offices.

In that image, we are looking for all paths between three offices of at most three concepts in between. We furthermore omit `rdf:type`, `cidoc:P2_has_type`, and `cidoc:P1_is_identified_by` relationships. This allows us to discover all the people that shared these offices. We furthermore clicked on one of the resources whose HTML representation is displayed on the right.

2.1 Generating Queries for Finding Paths

We want to avoid that a user has to wait for all results to appear. Rather than iteratively finding paths, we chose to generate queries for all possible paths, which is the focus of this section.

For each unique pair (non-ordered) of resources (x, y) selected in Fig. 1 (2), we generate $\left(2^{(d+1)} - 2\right)$ queries. We start with a simple triple pattern looking at relationships in both directions ($d = 1$). Then we concatenate these with additional triple patterns for intermediate concepts, also taking into account the different directions. For $d = 2$, we take the patterns that resulted from $d = 1$ and combined it with the two new patterns. That resulted in four additional patterns to be considered, totaling at six. This process is exemplified below.

```
(d=1) ?x0 ?pred1r ?x1 .
(d=1) ?x1 ?pred1l ?x0 .
(d=2) ?x0 ?pred1r ?x1 . ?x1 ?pred2r ?x2 .
(d=2) ?x0 ?pred1r ?x1 . ?x2 ?pred2l ?x1 .
(d=2) ?x1 ?pred1l ?x0 . ?x1 ?pred2r ?x2 .
(d=2) ?x1 ?pred1l ?x0 . ?x2 ?pred2l ?x1 .
(d=3) ?x0 ?pred1r ?x1 . ?x1 ?pred2r ?x2 . ?x2 ?pred3r ?x3 .
(d=3) ?x0 ?pred1r ?x1 . ?x1 ?pred2r ?x2 . ?x2 ?pred3l ?x3 .
(d=3) ?x0 ?pred1r ?x1 . ?x2 ?pred2l ?x1 . ?x2 ?pred3r ?x3 .
(d=3) ?x0 ?pred1r ?x1 . ?x2 ?pred2l ?x1 . ?x2 ?pred3l ?x3 .
(d=3) ...
```

We could have used property paths, but then we would have needed to keep track of the direction in the application. To avoid comparing URIs in `FILTER` clauses, we replace `?x0` by <*x*> and the largest variable `?x`*n* by <*y*> in each of the graph patterns. We do bind the URIs of x and y to these variables in `BIND` clauses. As we generate patterns for each unique pair, we cover all the possible graph patterns. This approach speeds up query evaluation.

We then introduce a couple of filters: none of the intermediate variables `?x`*i* should be bound to *x* or *y*; none of the intermediate variables `?x`*i* should be bound to the same resource to avoid cycles; none of the predicates should be bound to URIs appearing in the predicates to ignore.

All these patterns and filters were wrapped around a `SELECT DISTINCT *` and executed by the client. As for the number of queries generated, given n resources for which we want to explore the relations between them and d the maximum number of relations between those resources, the number of SPARQL queries generated is $\binom{n}{2}\left(2^{\{d+1\}} - 2\right)$. The tool uses the variable naming convention to keep track of the nodes and edges to be displayed. The 'l' and the 'r' in the variable names for predicates

is also used to determine the placement of the arrow marker on the edges.

3 Implementation

The code has been made available on GitHub with an accessible license.[1] The form relies on various JavaScript libraries (e.g., JQuery), and the visualization relies on D3.js. The tool uses JavaScript promises and callbacks to populate the visualizations as results come in. A limitation of our implementation is that indeed one is limited to the capabilities of one's computer, and visualizations may end up becoming too large or too cluttered. The latter, however, is a known challenge in visualization.

4 Conclusions

We presented Murmuration, a tool for visually exploring knowledge graphs in SPARQL endpoints. It is inspired by the state-of-the-art in Linked Data visualizations, though we compute the discovery of arbitrary paths between resources on the client-side. The approach is simple and indeed demanding resources on the client, but it does not scale for arbitrarily large values for d. We believe, however, this tool would come in handy to quickly explore, in a playful way, data contained in SPARQL endpoints.

The tool is made available with an accessible license and is currently used in a digital humanities project. Future work consists of looking into coping with blank nodes and federated querying, with priority given to the latter. We are also considering comparing small changes in the approach; e.g., halving the number of generated queries, and testing whether ?x0 is bound to <x> (or <y>) and ?xn is bound to <y> (or <x>).

Acknowledgements

Beyond 2022 is funded by the Government of Ireland, through the Department of Culture, Heritage and the Gaeltacht, under the Project Ireland 2040 framework. C. Debruyne is also partially supported by the ADAPT Centre for Digital Content Technology under the SFI Research Centres Programme (Grant 13/RC/2106).

References

1. Bikakis, N., Liagouris, J., Krommyda, M., Papastefanatos, G., Sellis, T.K.: graphvizdb: A scalable platform for interactive large graph visualization. In: 32nd IEEE International Conference on Data Engineering, ICDE 2016, Helsinki, Finland, May 16-20, 2016. pp. 1342–1345 (2016).
2. Bikakis, N., Sellis, T.K.: Exploration and visualization in the web of big linked data: A survey of the state of the art. In: Proceedings of the Workshops of the EDBT/ICDT 2016 Joint Conference, EDBT/ICDT Workshops 2016, Bordeaux, France, March 15, 2016 (2016), http://ceur-ws.org/Vol-1558/paper28.pdf

[1] https://github.com/chrdebru/murmuration

3. Heim, P., Hellmann, S., Lehmann, J., Lohmann, S., Stegemann, T.: Relfinder: Revealing relationships in RDF knowledge bases. In: Semantic Multimedia, 4th International Conference on Semantic and Digital Media Technologies, SAMT 2009, Graz, Austria, December 2-4, 2009, Proceedings. Lecture Notes in Computer Science, vol. 5887, pp. 182–187. Springer (2009).
4. Mouromtsev, D., Pavlov, D., Emelyanov, Y., Morozov, A., Razdyakonov, D., Galkin, M.: The simple web-based tool for visualization and sharing of semantic data and ontologies. In: Proceedings of the ISWC 2015 Posters & Demonstrations Track co-located with the 14th International Semantic Web Conference (ISWC-2015), Bethlehem, PA, USA, October 11, 2015 (2015), http://ceur-ws.org/Vol-1486/paper 77.pdf

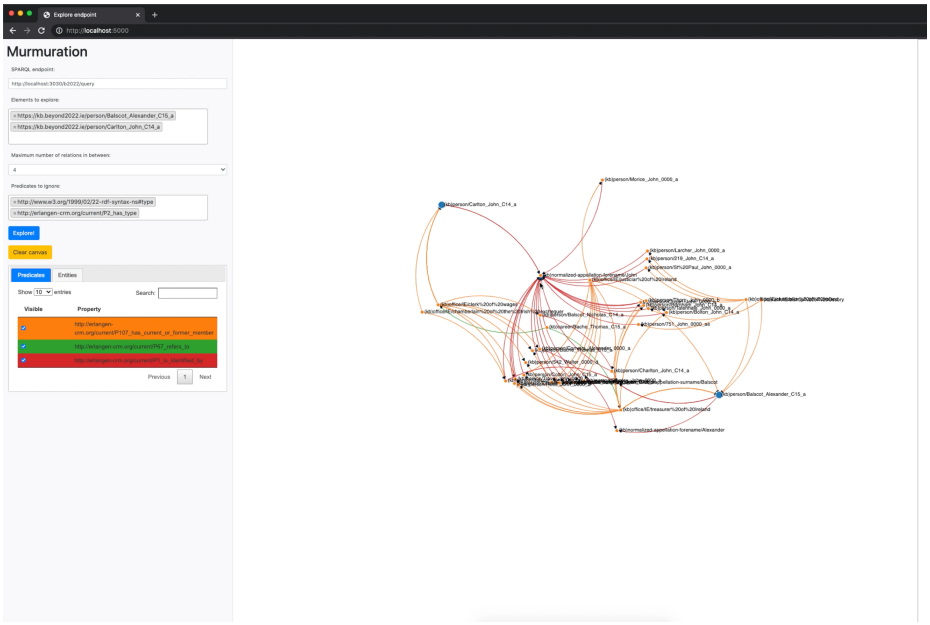

Fig. 2. Different relations have different colors in the graph, which can be hidden using the panel on the right.