A User Interface for Exploring and Querying Knowledge Graphs (Extended Abstract)*

Hernán Vargas[1,3†], **Carlos Buil-Aranda**[1,3], **Aidan Hogan**[2,3] and **Claudia López**[1]
[1]Universidad Técnica Federico Santa María, Valparaíso, Chile
[2]DCC, Universidad de Chile, Chile
[3]Millenium Institute for Foundational Research on Data (IMFD), Chile
{hvargas,cbuil,clopez}@inf.utfsm.cl, ahogan@dcc.uchile.cl

Abstract

As the adoption of knowledge graphs grows, more and more non-experts users want to explore and query such graphs. These users are often not familiar with graph query languages such as SPARQL, and may not be familiar with the knowledge graph's structure. We provide a summary of our work on a language and visual interface – called RDF EXPLORER – that helps non-expert users to navigate and query knowledge graphs. A usability study over Wikidata shows that users successfully complete more tasks with RDF EXPLORER than with the existing WIKIDATA QUERY HELPER interface.

1 Introduction

Knowledge graphs – which apply a graph-based abstraction to manage diverse data at large-scale [Hogan *et al.*, 2020] – have gained considerable attention recently. In this setting, knowledge graphs form a common substrate of knowledge within an enterprise or community [Noy *et al.*, 2019]. While enterprise knowledge graphs are used within companies such as eBay, Google, Facebook, IBM, Microsoft, etc. [Noy *et al.*, 2019]; open knowledge graphs, such as DBpedia [Lehmann *et al.*, 2015], Wikidata [Vrandečić and Krötzsch, 2014], etc., are published on the Web under liberal licenses.

In order to query knowledge graphs, a number of languages are now available, including SPARQL for RDF graphs, Cypher for property graphs, etc. [Angles *et al.*, 2017]. However, non-expert users that lack knowledge of these query languages, or of how the knowledge graph is structured, may struggle to express their information needs as queries.

Several systems have been proposed to help non-expert users explore and query knowledge graphs [Dadzie and Rowe, 2011]. These systems typically involve some combination of keyword search, faceted browsing, graphical browsing, query building, graph summarisation, visualisation techniques, and more besides. A general trade-off appears with respect to expressivity versus usability: more expressive systems support more complex forms of queries, but are often more difficult to use; for example, many systems, such as those based on faceted browsing, only support acyclic queries. Systems that do support cyclic queries often assume that users have some technical knowledge of query languages.

In this work, we propose a language and associated system – called RDF EXPLORER – that we claim helps users query knowledge graphs with more ease than existing interfaces. Our language is based on a simple set of operators that allows users to incrementally build (cyclic) graph pattern. Our system provides a visual user interface that supports these operators, as well as other features for enhancing usability, including auto-completion, result previews, etc. To evaluate our proposal, we conduct a task-based usability study comparing our interface with the WIKIDATA QUERY HELPER, where we can conclude, with statistical significance, that users achieve a higher successful completion rate of tasks with our system than the baseline. This paper is an extended abstract summarising system details and results that were previously published at ISWC 2019 [Vargas *et al.*, 2019].

2 Related Work

Various systems have been proposed to help non-expert users to visualise, explore and query graphs [Dadzie and Rowe, 2011]. Early works included *search systems* for finding nodes representing relevant entities by keyword [Berners-Lee *et al.*, 2006]. Other systems propose *visualisations* to help summarise and explore graphs [Sayers, 2004; Skjæveland, 2012]. To help non-experts pose complex queries over graphs, *query editors and builders* have also been investigated [Smart *et al.*, 2008; Hogenboom *et al.*, 2010; Clemmer and Davies, 2011; Grafkin *et al.*, 2016; Rietveld and Hoekstra, 2017]. These interfaces aim to help users with little expertise of the query language and/or data to express complex queries.

We propose a language for constructing graph patterns, and a query builder system that supports this language. Our system constructs SPARQL queries over RDF graphs, though it can readily adapted to other query languages over other graph models (e.g, Cypher over property graphs). In terms of previous works, SMEAGOL [Clemmer and Davies, 2011] is the closest to our proposal; however, a key difference is that SMEAGOL focuses on supporting tree-shaped queries generated during user exploration, whereas we support cyclic graph patterns. Our work is further distinguished by a systematic usability evaluation of our proposal.

*This paper is an extended abstract of a paper originally published at ISWC 2019 [Vargas *et al.*, 2019].

†Contact Author

3 RDF Explorer

We now summarise the details of our proposed language and system for helping non-experts to explore and query graphs.

3.1 Visual Query Graph

Our language is formulated with respect to a visual query graph. Let $\mathbf{I}$ denote the set of IRIs, $\mathbf{L}$ denote the set of literals and $\mathbf{V}$ denote the set of query variables. A *visual query graph* (VQG) is defined as a directed, edge-labelled graph $\mathsf{G} = (\mathsf{N}, \mathsf{E})$, with nodes N and edges E. The nodes of the VQG are a finite set of IRIs, literals and/or variables: $\mathsf{N} \subset \mathbf{I} \cup \mathbf{L} \cup \mathbf{V}$. The edges of the VQG are a finite set of triples, where each triple indicates a directed edge between two nodes with a label taken from the set of IRIs or variables: $\mathsf{E} \subset \mathsf{N} \times (\mathbf{I} \cup \mathbf{V}) \times \mathsf{N}$. We denote by $\text{var}(\mathsf{G})$ the set of variables appearing in $\mathsf{G} = (\mathsf{N}, \mathsf{E})$, either as nodes or edge labels: $\text{var}(\mathsf{G}) := \{v \in \mathbf{V} \mid v \in \mathsf{N} \text{ or } \exists\, n_1, n_2 : (n_1, v, n_2) \in \mathsf{E}\}$.

A VQG is constructed through a visual query language, consisting of a minimal set of four algebraic operators that will correspond to atomic user interactions: adding a variable node, adding a constant node, adding an edge between two existing nodes with a variable label, and adding an edge between two existing nodes with an IRI label. More specifically, as the base case, the VQG is initially empty: $\mathsf{G}_0 = (\emptyset, \emptyset)$. Thereafter, letting $\mathsf{G} = (\mathsf{N}, \mathsf{E})$ denote the current VQG; the *visual query language* (VQL) is defined recursively through the following four atomic operations: (1) *Initialise a new variable node*: $\eta(\mathsf{G}) := (\mathsf{N} \cup \{v\}, \mathsf{E})$ where $v \notin \text{var}(\mathsf{G})$. (2) *Add a new constant node*: $\eta(\mathsf{G}, x) := (\mathsf{N} \cup \{x\}, \mathsf{E})$ where $x \in (\mathbf{I} \cup \mathbf{L})$. (3) *Initialise a new edge between two nodes with a variable edge-label*: $\varepsilon(\mathsf{G}, n_1, n_2) := (\mathsf{N}, \mathsf{E} \cup \{(n_1, v, n_2)\})$ where $\{n_1, n_2\} \subseteq \mathsf{N}$ and $v \notin \text{var}(\mathsf{G})$. (4) *Add a new edge between two nodes with an IRI edge-label*: $\varepsilon(\mathsf{G}, n_1, x, n_2) := (\mathsf{N}, \mathsf{E} \cup \{(n_1, x, n_2)\})$ where $\{n_1, n_2\} \subseteq \mathsf{N}$ and $x \in \mathbf{I}$.[1]

VQGs can then be translated directly to basic graph patterns (BGPs) in the concrete (SPARQL) query syntax. If required, more expert users can then modify the resulting query. There are, however, some details of BGPs that cannot be expressed with the VQL, and indeed, some VQGs that cannot be translated to BGPs. In particular, we cannot use the VQL to express a join on an edge-label/predicate variable; such joins are rarely found in SPARQL query logs [Arias *et al.*, 2011] and their omission simplifies matters somewhat. On the other hand, "orphan nodes" without incident edges must be dropped when translating the VQL to a BGP. With respect to complexity, *query evaluation* of VQGs – deciding if a solution μ is a result for a query Q on a graph G – is tractable as projection is not supported [Pérez *et al.*, 2009].

3.2 The RDF Explorer Interface

The RDF EXPLORER system implements the VQL as an interface. Figure 1 shows how the main drawing pane is used to query Wikidata for movies directed by two siblings. The user also has access to two other panes (not shown) in the main application: a pane to the right allows for selecting a node

[1]For the operators $\eta(\mathsf{G})$ and $\varepsilon(\mathsf{G}, n_1, n_2)$, an arbitrary fresh variable can be automatically generated, where the resulting VQG will be unique modulo isomorphism on variables.

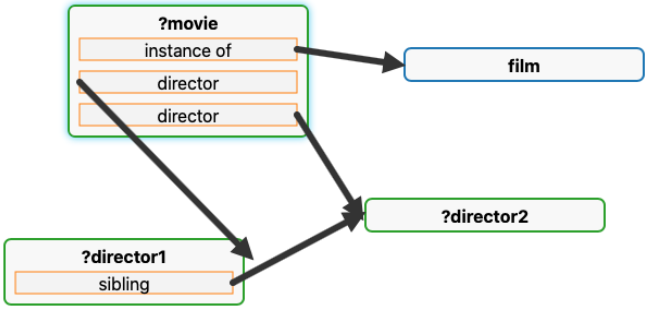


Figure 1: VQG for siblings who direct movies together

editor (allowing to restrict a highlighted node), a SPARQL query editor (showing the current query), and a help panel; a pane to the left provides a search panel.

The user is first presented with a blank visual query editor. The user then adds a new node, be it a variable ($\eta(\mathsf{G})$) or a constant ($\eta(\mathsf{G}, x)$). To find constants (entities) the user can enter a keyword query in the search pane to generate auto-completed results, any of the which can be dragged into the central query editor pane. The user may then proceed to add a second node by the same means. With two or more nodes available, the user can now click and drag between two nodes to generate an edge with a variable edge-label ($\varepsilon(\mathsf{G}, n_1, n_2)$). Clicking a variable edge-label will generate suggestions for IRIs that can replace the variable while ensuring that the query does not give empty results ($\varepsilon(\mathsf{G}, n_1, x, n_2)$).

The user can click on a variable node to view a sample of current results for that variable (generated by mapping the current VQG to SPARQL and projecting that variable). Clicking on a constant node, they will be shown the data for that node. Thus the user can explore the graph and receive feedback on the results generated thus far, guiding next steps [Bhowmick *et al.*, 2017]. Constant nodes can also be converted to variables nodes, allowing a user to explore a specific example and then generalise the graph to a graph pattern (i.e., a graph query) [Clemmer and Davies, 2011].

4 Usability Study

We now present a usability study intending to evaluate the RDF EXPLORER interface. We first enumerate our hypotheses and then describe the dataset, baseline system and study design used to test these hypotheses. We then characterise the participants of the study and give our main results.

4.1 Hypotheses

Our system aims to allow users without prior knowledge of graph query languages to explore and query a large, diverse knowledge graph. Our hypotheses cover different aspects relating to how the system succeeds in this goal relative to a baseline system. The first hypothesis covers success in terms of formulating a query graph that provides the expected answers when evaluated. The second and third hypotheses cover partial successes in terms of, respectively, formulating a query graph with some correct edges, and formulating a query graph with the same "shape" as a reference query graph (even if some constants in the query graph are incorrect).

H_1: *Non-expert users can correctly formulate more query graphs with our system than a baseline system.* For a query graph to be considered correct, it must return the same results as the reference query for the task.

H_2: *Non-expert users are able to correctly formulate more edges with our system than a baseline system.* For an edge (i.e., *triple pattern*) in a query graph to be considered correct, it must be contained in the reference query (modulo variable names).

H_3: *Non-expert users are able to generate more query graphs with the correct structure using our system than a baseline system.* For a query graph to be considered as having the correct structure, it must have the same "shape" as the reference query graph, irrespective of the constants used. More formally, given a VQG $\mathsf{G} = (\mathsf{N}, \mathsf{E})$, let $\mathrm{shape}(\mathsf{G})$ denote a directed graph $S = (V_S, E_S)$ such that $V_S = \mathsf{N}$ and $(x, y) \in S$ if and only if there exists an edge-label l such that $(x, l, y,) \in \mathsf{E}$; now given the reference query graph G', a user's query graph G'', and their corresponding shapes $S' = \mathrm{shape}(\mathsf{G}')$ and $S'' = \mathrm{shape}(\mathsf{G}'')$, the user's query graph G'' is considered correct if and only if there exists an isomorphism $h : V''_S \to V'_S$ such that $h(S'') = S'$.

4.2 Knowledge Graph and Baseline System

We chose Wikidata as the knowledge graph for our evaluation; this knowledge graph currently describes 80 millions items with thousands of different properties. We chose the official WIKIDATA QUERY HELPER [Malyshev *et al.*, 2018] (WQH) as our baseline system; it provides a form with autocompletion alongside a form for editing the concrete SPARQL syntax of the query, where changes in one are reflected in the other.[2] Given that Wikidata receives tens of thousands of "organic" queries per day [Malyshev *et al.*, 2018], the baseline WQH system is likely to be widely used by people with varying expertise of SPARQL and Wikidata.

4.3 Study Design

To test our hypotheses, we conduct a task-based user study to compare the participants' ability to solve tasks on the proposed interface versus the baseline interface [Munzner, 2014]. We use a within-subject design where each participant completes five tasks using our query builder and five similar tasks with the baseline. Each task consists of answering a question (given in natural language) that requires formulating a query to retrieve answer(s) from the Wikidata graph.

We divide the subjects into two groups. The first group is given a set of five tasks, where each task requires formulating a query (S1: *find all dogs*, *find all popes who are female*, *find all mountains located in European countries*, *find all emperors whose father is also an emperor* and *find all Nobel prize winners with a student who won the same Nobel prize*) using the proposed interface; afterwards they are asked to formulate an analogous set of five queries (S2: *find all actors*, *find all German soccer players who participated in FIFA 2014*, *find all container ships located in European countries*, *find all physicists whose spouse is also a physicist*, and *find all relatives participating in the same Olympic sport*) but this time using the baseline. Conversely, the second group is asked to build the first set of queries (S1) using the baseline and the second set (S2) with the proposed interface.

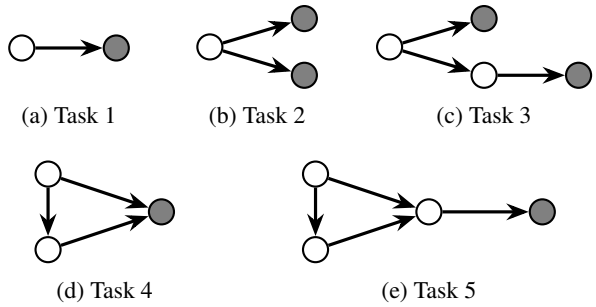


Figure 2: Reference query shapes for the five tasks of each set

This design controls for individual differences by having participants use both interfaces. Mixing the order of interfaces in both groups helps to control for carry-over effects, such as learning or fatigue. The queries in both sets are designed to be of increasing difficulty to follow a learning curve and also to avoid users being discouraged early on. We further aim to keep the n^{th} query of both sets of tasks comparable in terms of difficulty. As shown in Figure 2, each pair of tasks corresponds to the same abstract visual query graph, and each successive pair incrementally adds more complexity.

Before each set of tasks, we briefly trained the participants on how to use the interface they were about to see (participants had to solve the task *"Find all Clint Eastwood movies in which any of his children participated"* using the respective interface). A web page with other example queries for the interfaces was also provided to the participants.

4.4 Study Participants

The study was conducted with 28 students enrolled in the undergraduate course "*User interface design*". The students were in the fourth year of a Computer Science undergraduate program in a Spanish-speaking university. They were not familiar with SPARQL nor with Wikidata. Participants were given up to 40 minutes to solve each of the sets of five tasks using each interface; adding two 5 minute tutorials before both sets of tasks, the total study time was thus 90 minutes.

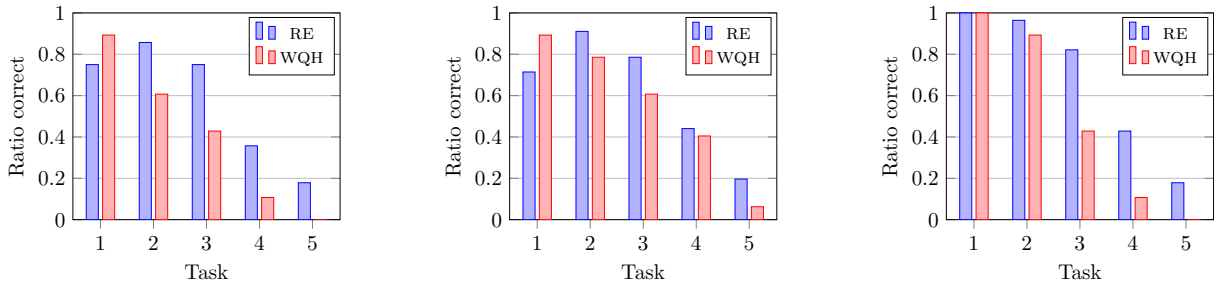
4.5 Metrics

To compare our proposal to the baseline, we measure diverse aspects of the users' ability to perform the give tasks, including task success rate and time for task completion. We also store the query graphs to assess (partial) correctness.

5 Results

We now present the ratio of correct responses broken down by three levels of granularity – queries, edges and shapes – and then test the corresponding hypotheses.

[2] Available from http://query.wikidata.org/. At the time of writing, the form-based query editor of WQH that was used for experiments has been removed. The details of this query editor are described by Malyshev *et al.* [2018].



(a) Queries (b) Triple patterns (c) Shapes

Figure 3: Mean ratios of correct results at three levels of granularity

5.1 Ratio of Correct Responses

Figure 3 shows the mean ratios of correct responses for the proposed interface (RDF EXPLORER = RE) and the baseline interface (WIKIDATA QUERY HELPER = WQH) at three different levels: queries, edges, and shapes. While queries and shapes are either correct or incorrect for a response, in the case of edges, we take the ratio of correct edges to total edges in a response and present the mean of these ratios. In the first task, which involves a single edge, WQH obtains better results than RE. However, as queries become more "graph-like", results are better for RE than WQH. Comparing the three levels of granularity, in the RE system, users generally have more success defining the correct query graph shape than identifying the terms (constants) in the query graph; the opposite trend is true for WQH, where users can more easily find the correct query terms, but not the correct shape. A possible explanation is that WQH is form-based while RE is graph-based; another factor is that RE prevents users from creating query shapes that give empty results while WQH does not.

5.2 Hypothesis Testing

To assess our hypotheses, we use *paired-t* tests to compare the success rates (depicted in Figure 3) of participants using both tools. We use $\alpha = 0.05$ to reject the null hypothesis. Given $n = 28$, we say that we have obtained a significant result when t^* is greater than $t_{crit} = 2.052$ ($t^* \geq t_{crit}$). For our three hypotheses (see Section 4) the null hypothesis is that there is no difference between the tools or that WQH performs better. The alternative hypothesis is that RE performs better. We denote the success rates for RE as $\bar{x}$ and those for WQH as $\bar{y}$; the average distances we denote by $\bar{d} = (\bar{x} - \bar{y})$, and the standard deviation by s_d. We now test the three hypotheses:

H_1 : *Non-expert users can correctly formulate more query graphs with our system than a baseline system.* We use the raw data of Figure 3a. With $\bar{d} = \overline{(\bar{x} - \bar{y})} = 0.1714$ and $s_d = 0.2813$ we obtain $t^* = 3.22 > t_{crit} = 2.052$ rejecting the null hypothesis: we validate H_1 with a statistically significant result favouring our interface.

H_2 : *Non-expert users are able to correctly formulate more edges with our system than a baseline system.* We use the raw data of Figure 3b. With $\bar{d} = 0.06$ and $s_d = 0.2609$ we obtain $t^* = 1.1947 < t_{crit} = 2.052$: the results are not statistically significant.

H_3 : *Non-expert users are able to generate correctly structured query graphs with our system than a baseline system.* We use the raw data of Figure 3c. With $\bar{d} = 0.1928$ and $s_d = 0.2801$ we obtain $t^* = 3.6431 > t_{crit} = 2.052$ rejecting the null hypothesis: we validate H_3 with a statistically significant result favouring our interface.

Our study is thus conclusive regarding the claim that our proposed interface is better than the baseline at helping non-expert users formulate their queries as graphs (H_1, H_3), but is inconclusive regarding the claim of our interface being better at helping users to correctly generate edges (H_2).

5.3 Other Results

We also posed questionnaires to the participants to assess their subjective opinions of the two interfaces. The results show that users preferred RE in all subjective criteria we analysed except physical effort (RE requires more use of the mouse than WQH) and perceived performance (RE generates result previews and more precise suggestions, which may cause lag). We refer to the ISWC 2019 paper [Vargas *et al.*, 2019] for details of these results, as well as task times.

6 Conclusions

We propose a language and visual interface to help non-expert users formulate queries over knowledge graphs. Our results indicate that our interface is, in general, more usable than the baseline system used by Wikidata. In terms of aspects to improve, lower success rates for later tasks suggest that non-expert users still struggle to express more complex queries. More expressive features of SPARQL (e.g., aggregations, unions, etc.) could be supported. The system also generates some lag when generating suggestions that could be improved by approximation techniques.

Live demos of the RDF Explorer system are available online for the Wikidata (https://www.rdfexplorer.org/) and DBpedia (https://dbpedia.rdfexplorer.org/) knowledge graphs.

Acknowledgements

Vargas and Buil-Aranda were supported by Fondecyt Iniciación Grant No. 11170714. Hogan was supported by Fondecyt Grant No. 1181896. Vargas, Buil-Aranda and Hogan were supported by the Millennium Institute for Foundational Research on Data (IMFD).

References

[Angles *et al.*, 2017] Renzo Angles, Marcelo Arenas, Pablo Barceló, Aidan Hogan, Juan L. Reutter, and Domagoj Vrgoc. Foundations of Modern Query Languages for Graph Databases. *ACM Comput. Surv.*, 50(5):68:1–68:40, 2017.

[Arias *et al.*, 2011] Mario Arias, Javier D. Fernández, Miguel A. Martínez-Prieto, and Pablo de la Fuente. An Empirical Study of Real-World SPARQL Queries. In *Usage Analysis and the Web of Data (USEWOD)*, 2011.

[Berners-Lee *et al.*, 2006] Tim Berners-Lee, Yuhsin Chen, Lydia Chilton, Dan Connolly, Ruth Dhanaraj, James Hollenbach, Adam Lerer, and David Sheets. Tabulator: Exploring and analyzing linked data on the semantic web. In *Proceedings of the 3rd international semantic web user interaction workshop*, volume 2006, page 159. Citeseer, 2006.

[Bhowmick *et al.*, 2017] Sourav S Bhowmick, Byron Choi, and Chengkai Li. Graph querying meets HCI: State of the art and future directions. In *ACM International Conference on Management of Data*, pages 1731–1736. ACM, 2017.

[Clemmer and Davies, 2011] Aaron Clemmer and Stephen Davies. Smeagol: a "specific-to-general" semantic web query interface paradigm for novices. In *Database and Expert Systems Applications (DEXA)*, pages 288–302. Springer, 2011.

[Dadzie and Rowe, 2011] Aba-Sah Dadzie and Matthew Rowe. Approaches to visualising Linked Data: A survey. *Semantic Web*, 2(2):89–124, 2011.

[Grafkin *et al.*, 2016] Pavel Grafkin, Mikhail Mironov, Michael Fellmann, Birger Lantow, Kurt Sandkuhl, and Alexander V Smirnov. Sparql query builders: Overview and comparison. In *BIR Workshops*, 2016.

[Hogan *et al.*, 2020] Aidan Hogan, Eva Blomqvist, Michael Cochez, Claudia d'Amato, Gerard de Melo, Claudio Gutierrez, José Emilio Labra Gayo, Sabrina Kirrane, Sebastian Neumaier, Axel Polleres, Roberto Navigli, Axel-Cyrille Ngonga Ngomo, Sabbir M. Rashid, Anisa Rula, Lukas Schmelzeisen, Juan F. Sequeda, Steffen Staab, and Antoine Zimmermann. Knowledge graphs. *CoRR*, abs/2003.02320, 2020.

[Hogenboom *et al.*, 2010] Frederik Hogenboom, Viorel Milea, Flavius Frasincar, and Uzay Kaymak. RDF-GL: A SPARQL-Based Graphical Query Language for RDF. In *Emergent Web Intelligence: Advanced Information Retrieval*, pages 87–116. 2010.

[Lehmann *et al.*, 2015] Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N. Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick van Kleef, Sören Auer, and Christian Bizer. Dbpedia - A large-scale, multilingual knowledge base extracted from Wikipedia. *Semantic Web*, 6(2):167–195, 2015.

[Malyshev *et al.*, 2018] Stanislav Malyshev, Markus Krötzsch, Larry González, Julius Gonsior, and Adrian Bielefeldt. Getting the Most Out of Wikidata: Semantic Technology Usage in Wikipedia's Knowledge Graph. In *International Semantic Web Conference (ISWC)*, pages 376–394. Springer, 2018.

[Munzner, 2014] Tamara Munzner. *Visualization analysis and design*. AK Peters/CRC Press, 2014.

[Noy *et al.*, 2019] Natalya Fridman Noy, Yuqing Gao, Anshu Jain, Anant Narayanan, Alan Patterson, and Jamie Taylor. Industry-scale knowledge graphs: lessons and challenges. *Commun. ACM*, 62(8):36–43, 2019.

[Pérez *et al.*, 2009] Jorge Pérez, Marcelo Arenas, and Claudio Gutiérrez. Semantics and complexity of SPARQL. *ACM Trans. Database Syst.*, 34(3):16:1–16:45, 2009.

[Rietveld and Hoekstra, 2017] Laurens Rietveld and Rinke Hoekstra. The YASGUI family of SPARQL clients. *Semantic Web*, 8(3):373–383, 2017.

[Sayers, 2004] Craig Sayers. Node-centric rdf graph visualization. *Mobile and Media Systems Laboratory, HP Labs*, 2004.

[Skjæveland, 2012] Martin G Skjæveland. Sgvizler: A javascript wrapper for easy visualization of sparql result sets. In *Extended Semantic Web Conference*, pages 361–365. Springer, 2012.

[Smart *et al.*, 2008] Paul R. Smart, Alistair Russell, Dave Braines, Yannis Kalfoglou, Jie Bao, and Nigel R. Shadbolt. A Visual Approach to Semantic Query Design Using a Web-Based Graphical Query Designer. In *Knowledge Engineering and Knowledge Management (EKAW)*, pages 275–291. Springer, 2008.

[Vargas *et al.*, 2019] Hernán Vargas, Carlos Buil Aranda, Aidan Hogan, and Claudia López. RDF Explorer: A Visual SPARQL Query Builder. In *International Semantic Web Conference (ISWC)*, pages 647–663, 2019.

[Vrandečić and Krötzsch, 2014] Denny Vrandečić and Markus Krötzsch. Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85, 2014.