LDViz: a tool to assist the multidimensional exploration of SPARQL endpoints

Aline Menin, Pierre Maillot, Catherine Faron, Olivier Corby, Carla Maria Dal Sasso Freitas, Fabien Gandon, Marco Winckler

LDViz: a tool to assist the multidimensional exploration of SPARQL endpoints

Aline Menin[1][0000−0002−9345−3994], Pierre Maillot[1][0000−0002−9814−439X], Catherine Faron[1][0000−0001−5959−5561], Olivier Corby[1][0000−0001−6610−0969], Carla Dal Sasso Freitas[2][0000−0003−1986−8435], Fabien Gandon[1][0000−0003−0543−1232], and Marco Winckler[1][0000−0002−0756−6934]

[1] University Côte d'Azur, CNRS, Inria, I3S (UMR 7271), France
{aline.menin, pierre.maillot, catherine.faron, olivier.corby, fabien.gandon, marco.winckler}@inria.fr
[2] Institute of Informatics, Federal University of Rio Grande do Sul, Porto Alegre, Brazil
carla@inf.ufrgs.br

Abstract. Over recent years, we witnessed an astonishing growth in production and consumption of Linked Data (LD), which contains valuable information to support decision-making processes in various application domains. In this context, data visualization plays a decisive role in making sense of the large volumes of data created every day and in effectively communicating structures, processes, and trends in data in an accessible way. In this paper, we present LDViz, a visualization tool designed to support the exploration of knowledge graphs via multiple perspectives: (i) RDF graph/vocabulary inspection, (ii) RDF summarization, and (iii) exploratory search. We demonstrate the usage and feasibility of our approach through a set of use case scenarios showing how users can perform searches through SPARQL queries and explore multiple perspectives of the resulting data through multiple complementary visualization techniques. We also demonstrate the reach and generic aspects of our tool through an evaluation that tests the support of 419 different SPARQL endpoints.

Keywords: Linked Data · Linked Data Visualization · RDF Visualization · Visual Exploratory Search · SPARQL Endpoints Exploration

1 Introduction

An increasing amount of data is published as RDF (Resource Description Framework) datasets and is made available as Linked Open Data (LOD) in different domains, providing valuable information to support decision-making processes in various application domains [15]. However, the value of these data depends on the ability of decision makers to grasp the relevant information to describe the phenomena embedded in the data. Information visualization, through the use of visual representations of abstract data, reinforces human cognition to support the discovery of unstructured insights only limited by human imagination and creativity, making it a suitable approach to communicate the knowledge described by RDF datasets. In particular, we observe an increasing interest in using visual and interactive techniques to explore LOD resources via multiple criteria and levels of abstraction by the Semantic Web community to accomplish three main goals: (i) to explore the relevant concepts of an application domain via ontology representation; (ii) to inspect RDF graphs (e.g., "for debugging triples") [1]; and (iii) to analyze the instances based on their types/classes.

Among the many evolution trends of the Web, knowledge graphs (KGs) are now widely used to describe in standard ways the semantics of entities in the real world and their relations [16], and to link descriptions with additional information in semantic LOD repositories. Typically, KGs are generated through the integration of many different data sources, which results on highly heterogeneous information. This heterogeneity represents both a leverage and a challenge in their effective utilization. Further to the often unknown structure and nature of the data, visualizing linked data requires a preceding KG processing to retrieve suitable data, which requires knowledge of the underlying RDF vocabulary used to build the KG, less and less familiar even to data producers and analysts, as different vocabularies can be used to describe the same phenomenon, and nearly inaccessible to application domain users. Furthermore, retrieving suitable data often requires combining data from different KGs (available from the same or different SPARQL endpoints), which results in several data quality issues (e.g., missing data, inconsistency, etc.). Thus, visual methods are a necessary and suitable approach to support an effective exploration of knowledge graphs.

The design process of every visualization tool follows a well-known pipeline (i.e., import $\rightarrow$ transform $\rightarrow$ map $\rightarrow$ render $\rightarrow$ interact) [4, 31]. In particular, a visualization pipeline for LOD data should also take into account the linked nature of these datasets by leveraging/supporting/exploiting these links while being capable of processing and visualizing the data appropriately. In a previous work [25], we presented and discussed a visualization pipeline for LOD exploration that supports a high level of flexibility in every step. This versatility is found in the drafting of SPARQL queries in a way that appropriately addresses the links in the linked data, in the possibility of tuning the parameters of the graphic display and the associated interaction, and in the availability of multiple visualization techniques that can help users see data according to diverse and complementary viewpoints. To demonstrate the feasibility of our visualization pipeline, we had implemented a proof of concept in the form of a web-based visualization tool called LDViz. In this paper, we further explore the genericity and flexibility of LDViz by defining a scope of SPARQL queries to support the exploration of RDF graphs via different methods and by evaluating the extent to which LDViz can support LOD visualization. In particular, our contributions are summarized as follows:

– A generic web-based visualization tool for LOD exploration, LDViz, that supports data visualization through multiple perspectives from any SPARQL endpoint that is W3C compliant.

- A classification of the scope of SPARQL queries with respect to KGs exploration methods. This classification cover RDF graph/vocabulary inspection, RDF summarization, and exploratory search.
- An analysis of LDViz using 419 SPARQL endpoints, which results shows an average coverage of 41.77% of SPARQL endpoints by our approach.

The remaining of this document is organized as follows. Section 2 presents the proposed visualization tool. Section 3 presents the scope of SPARQL queries in terms of KG exploration methods and illustrate their use in LDViz. Section 4 presents a coverage analysis of the genericity and reach of our approach. Section 5 summarizes previous contributions for LOD visualization and compares them with our approach. Section 6 discusses our results and concludes the paper.

2 Visual Exploration of LOD

In this section, we present the Linked Data Visualizer (LDViz), a web-based visualization tool for LOD exploration. Our visualization techniques are implemented using *D3.js* (Data-Driven Documents) library, while the *nodejs* library is used to manage the linked data access server that handles data retrieval through SPARQL queries. We also use *Stencil JS* to implement visualization techniques as reusable Web components. LDViz implements each step of the visualization pipeline (i.e., import → transform → map → render → interact) as described in [25] and summarized hereafter:

Import. Data import is handled via SPARQL queries. The generality of LDViz relies on the fact that users can query any SPARQL endpoint as long as it can return result sets in a JSON format. We provide an interactive interface where the user can test and debug SPARQL queries or import predefined queries, which they may modify at will. The data import process can be launched at different times throughout the exploration process by using follow up queries, which allows to import external data (a different subset of data from the same SPARQL endpoint or data from a different SPARQL endpoint) into the exploration process to enrich the analysis (bring supplementary information to the analysis or compare datasets).

Transform. Data transformation occurs in three moments during the exploration process. First, at the definition of the SPARQL query, the RDF graph is filtered to retrieve the appropriate data to solve a particular domain question and reshaped into the required data model (see Subsection 2.1) to be visualized. Second, in the transformation engine, the SPARQL result sets are cleaned and re-shaped into a suitable data model for visualization, handled by MGExplorer [23], a visualization interface to explore multidimensional network data. Finally, as the user filters the input data set through a selection operation in a particular view to explore it in another, the data are filtered and reshaped to fit the selected visualization technique.

Visual Mapping. Visual mapping occurs during the transformation of the SPARQL results set into the LDViz data model, followed by the mapping of data variables into the visual variables of each technique, and the tuning of certain variables through the use of a Graph Style Sheet (e.g., by defining colors to represent them) (see Subsection 2.2).

Rendering. This is handled by MGExplorer [23], the visualization interface to explore multidimensional network data mentioned before.

Interaction. Via the MGExplorer interface, we provide selection operations that allow the user to subset the input data to be explored using different visualization techniques, which present complementary views of the data.

2.1 Importing data from SPARQL endpoints

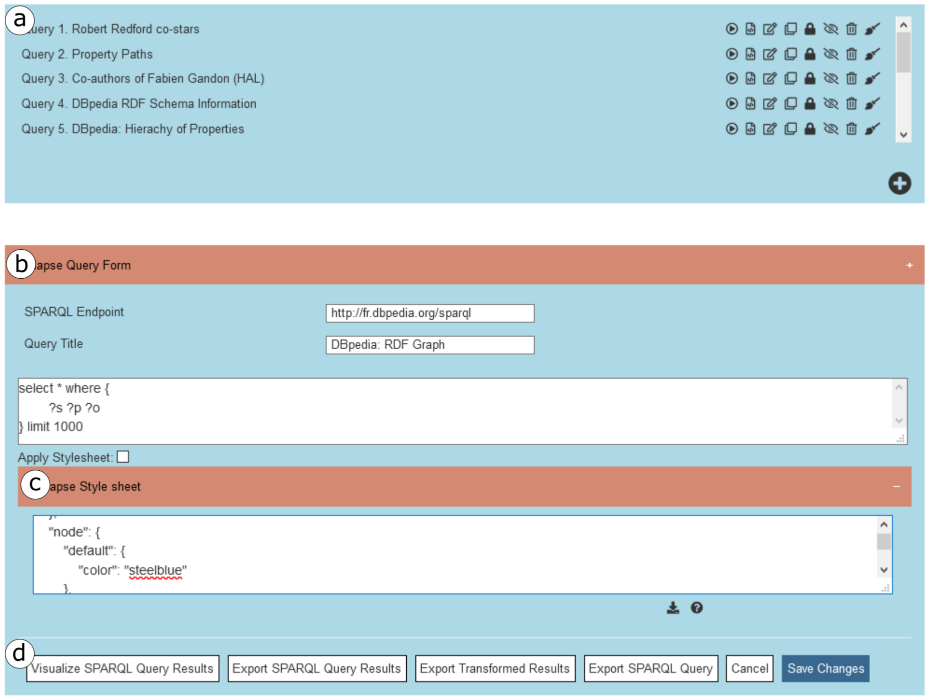
SPARQL Result Sets The W3C Recommendation [29] describes a specific data format to represent SPARQL SELECT query results using JSON. The results of a SPARQL query are serialized in a single top-level JSON object with two keys: `head` and `results`. The `results` key is an object with a single key, `bindings`, which is an array with zero or more elements, one element per query solution. The Listing 1.1 illustrates this data format through an extract of the results of the SELECT query presented in Listing 1.5. Each SPARQL query solution is a JSON object whose keys are the variable names of the query solution. A solution describes an RDF term that has a `type` and a `value` key, and other keys depending on the specific kind of RDF term (e.g., language, datatype). In LDViz, we use this data format, which means that our approach supports data from any SPARQL endpoint, as long as it can return SPARQL result sets in a JSON format that is W3C compliant.

```
{head: { link: [], vars: [ "s", "p", "o", "label", "type", "date" ] },
 results: { distinct: false, ordered: true, bindings: [
    {s: { type: "literal", xml:lang: "en", value: "Maximilian Schell" },
    p: { type: "uri",
        value: http://dbpedia.org/resource/A_Bridge_Too_Far_(film)},
    o: { type: "literal", xml:lang: "en", value: "Dirk Bogarde"},
    label: { type: "literal", xml:lang: "en",
        value: "A Bridge Too Far (film)"},
    type: { type: "literal", xml:lang: "en", value: "non-fiction" },
    date: { type: "typed-literal",
        datatype: http://www.w3.org/2001/XMLSchema#date,
        value: "1977-06-15" }}
  ] } }
```

Listing 1.1. Example of a SPARQL SELECT result set serialized in a JSON object as specified by the W3C Recommendation.

LDViz Data Model The data model corresponds to a custom graph model defined through a SPARQL SELECT query, which uses arbitrary query patterns on RDF graphs to generate the edges `?s ?p ?o` of the graph that one wants to visualize, where `?s` and `?o` represent the nodes of the graph while `?p` corresponds to labeled edges between them. Listing 1.5 illustrates an example SPARQL query supported by LDViz. In this example, `?s` and `?o` are bound to the actors and `?p` to the films. The result of this SPARQL query will be used to build a visualization of the social network of actors co-starring in films. In addition to these three variables, the data model allows three other reserved variables to be used to describe the edges (`?p`) of the output graph in visualization: `?type`, `?label`, and `?date`. Variable `?type` can be used to type the edges of the output graph (e.g., in a graph where films connect actors, films can be “typed” or classified by their genre). Due to human perceptual and cognitive limits towards visualizations, only a certain number of graphic elements can be perceived on the screen. For that, we allow the variable `?type` to be bound to only four different values that describe the edges. If the variable `?type` is bound to more than four distinct values in the SPARQL query result, the system automatically determines the three more relevant ones based on the number of bindings and considers the remaining values as the ”Other” category. The variable `?label` is intended to provide a description of the edges in natural language (e.g., the value of properties `rdfs:label` that describe resources). Finally, the `?date` variable is used to provide a visual representation of the distribution of edges over time (e.g., if edges are films, it could correspond to the release year).

2.2 SPARQL Query Editor



uery 1. Robert Redford co-stars
Query 2. Property Paths
Query 3. Co-authors of Fabien Gandon (HAL)
Query 4. DBpedia RDF Schema Information
Query 5. DBpedia: Hierachy of Properties

apse Query Form

SPARQL Endpoint	http://fr.dbpedia.org/sparql
Query Title	DBpedia: RDF Graph

```
select * where {
      ?s ?p ?o
} limit 1000
```

Apply Stylesheet: ☐

apse Style sheet

```
  "node": {
     "default": {
        "color": "steelblue"
     }
```

Visualize SPARQL Query Results | Export SPARQL Query Results | Export Transformed Results | Export SPARQL Query | Cancel | Save Changes

Fig. 1. SPARQL Query Management Interface. (a) Listing of predefined queries. (b) The querying area. (c) The GSS editing area. (d) Control buttons to visualize and export the results. Image reused from [25].

The query editor (Fig. 1) allows users to create, test, and debug SPARQL queries. Users can also clone predefined queries and adapt them according to specific needs[3]. The interface expects a SPARQL endpoint, a name for the SPARQL query and the query code itself. Users can retrieve data from more than one endpoint by leveraging the full strength of the SPARQL language, including the SERVICE clause by using the Corese proxy [9], for example. The action buttons at the bottom (Fig. 1d) allow to visualize the SPARQL query results using MGExplorer or export them as a JSON file.

```
(a) {"node": { "fst": {"color": "green"},
    "snd": {"color": "orange"} },
    "services": { "Corese Browser": { "url":
    "http://corese.inria.fr/srv/service/covid?uri="}}}

(b) select * where { ?s ?p ?o
    bind("fst" as ?style1) bind("snd" as ?style2)}
```

Listing 1.2. Example of (a) GSS and (b) its usage in a SPARQL query

Each query is associated with a **Graph Style Sheet (GSS)** that can be used to transform the default node-link diagram through a declarative specification of visibility, layout, and styling rules [28]. So that, it is possible to define styling rules as classes in a style sheet of reference (JSON format) (e.g., Listing 1.2a) and bind them to dedicated variables in the SPARQL query (i.e. `?style1` to style `?s`, `?style2` to style `?o`, and `?style` for both). This information is then processed in the transformation engine, which associates the style classes to the visual variables used in the visualization. Moreover, the GSS supports a behavior feature that enables exploring data (the graph nodes) via an external service (e.g., the Corese browser [9], which allows browsing the original repository of open data) as long as an URL is provided (see Listing 1.2a).

2.3 Using predefined queries

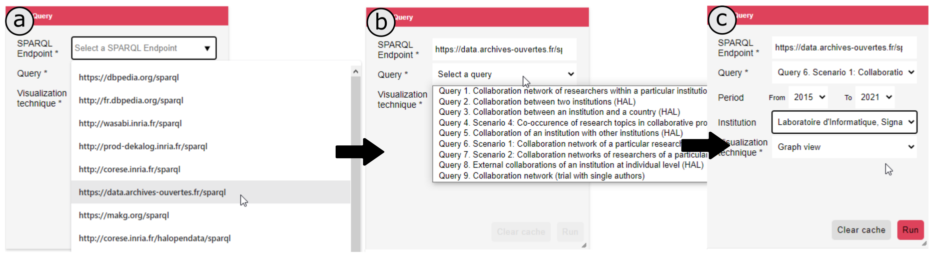

a
Query
SPARQL Endpoint *
Select a SPARQL Endpoint
Query *
Visualization technique *
https://dbpedia.org/sparql
http://fr.dbpedia.org/sparql
http://wasabi.inria.fr/sparql
http://prod-dekalog.inria.fr/sparql
http://corese.inria.fr/sparql
https://data.archives-ouvertes.fr/sparql
https://makg.org/sparql
http://corese.inria.fr/halopendata/sparql
b
Query
SPARQL Endpoint *
https://data.archives-ouvertes.fr/s
Query *
Select a query
Visualization technique *
Query 1. Collaboration network of researchers within a particular institutio
Query 2. Collaboration between two institutions (HAL)
Query 3. Collaboration between an institution and a country (HAL)
Query 4. Scenario 4: Co-occurence of research topics in collaborative pro
Query 5. Collaboration of an institution with other institutions (HAL)
Query 6. Scenario 1: Collaboration network of a particular researc
Query 7. Scenario 2: Collaboration networks of researchers of a particula
Query 8. External collaborations of an institution at individual level (HAL)
Query 9. Collaboration network (trial with single authors)
Clear cache
Run
c
Query
SPARQL Endpoint *
https://data.archives-ouvertes.fr/s
Query *
Query 6. Scenario 1: Collaboratio
Period
From 2015
To 2021
Institution
Laboratoire d'Informatique, Signa
Visualization technique *
Graph view
Clear cache
Run

Fig. 2. Using a query panel, users can (a) choose a SPARQL endpoint they want to explore, (b) a predefined query to start the exploration process, and (c) custom certain parameters of the query such as time period, location, etc.

[3] For security reasons, authentication is required to use the editor. Interested readers might contact the authors to acquire access.

We assume that many users might be expert on the application domain of an endpoint and interested in exploring such as data sets. However, an expertise in the application domain does not imply that the user knows SPARQL. For that kind of user, the visualization tool includes a querying process that allows the use of predefined queries (defined by expert users in the SPARQL query editor) to retrieve data from endpoints without having to understand SPARQL or the complexity of the underlying knowledge graph. From a query panel (see Fig. 2) users can select a SPARQL endpoint (Fig. 2a) and have a simple access to the queries (Fig. 2b) that have been specifically created for that endpoint. That panel also displays a set of custom parameters that allow users to filter the data (e.g. in a bibliometric network, these could be the publication period and research institution of scholarly articles) (Fig. 2c). The button "Run" at the bottom of the panel, will trigger the query against the chosen endpoint, prompt the system to transform the resulting data, and then launch the visualization technique to display the resulting data. For the purpose of optimizing the process, we use a cache that stores the results of queries for a certain amount of time (i.e. 15 days), thus reducing the requests to the data server. To acquire fresh data, the user can deliberately clear the results stored in the cache by using the button "Clear cache" at the bottom, which will force the system to apply the query to the SPARQL endpoint at the next execution.

2.4 Data Visualization using MGExplorer

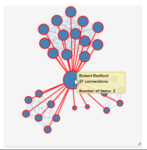
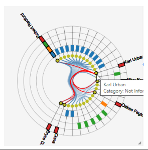
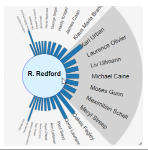
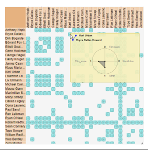
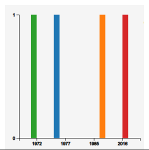
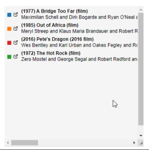
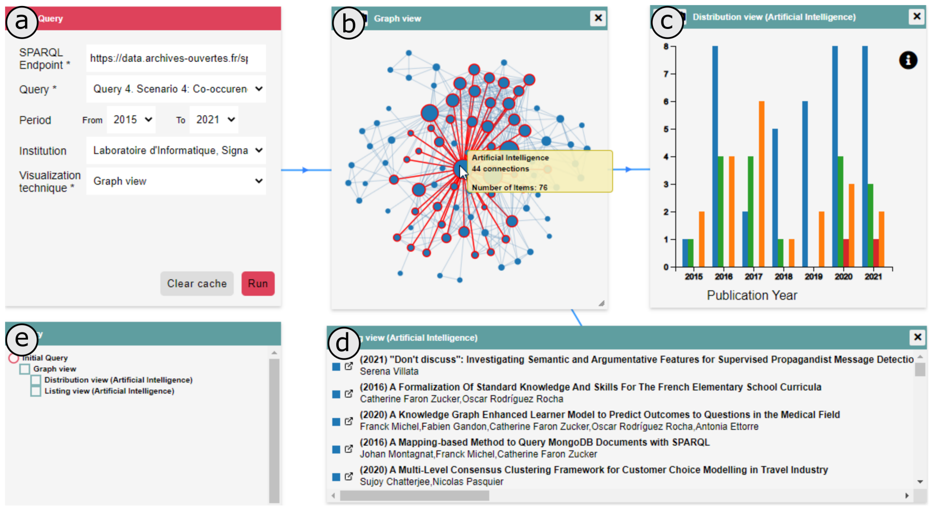
Graph View	Cluster View	Egocentric View	Pairwise Relationship View	Distribution View	Listing View
					

Table 1. Visualization techniques available in MGExplorer according to the given perspective to the data.

In our approach, data visualization is provided through MGExplorer [23], a tool that assists in the exploration of multidimensional and multivariate graphs. The tool provides a set of complementary visualization techniques (see Table 1) that can be instantiated at will during the exploration process to further explore the data through different perspectives. The **graph view** shows the nodes as items and the edges between them as relationships. This visualization provides an overview of the network defined by the SPARQL query. The **cluster view** [6] shows clusters according to some relationship among the data items. The technique features a multi-ring layout, where the innermost ring is formed by the data items (represented by circles), and the remaining rings display the data attributes (represented by rectangles). The items belonging to the same cluster are connected via curved lines. The **egocentric view** isolates a data item of interest (in the center) and shows all other data items with which it has a specific relationship in a circular view [7]. The data attributes of the pairwise relationships are encoded by the height and color of a bar placed between the item of interest and each related item. The user can place any item in the field of view center by clicking on it, switching the focus of the IRIS. The **pairwise relationship view** [5] features a matrix in which rows and columns represent data items, and cells contain glyphs that encode attributes that describe the relationship between these items. The default glyph is a star-plot-shaped object with a variable number of axes that are used to encode the values of the selected data attributes. By pointing a glyph to the matrix, it is possible to enlarge the glyph to see the details of the data attributes. The **distribution view** shows the data attributes of an item or a set of items distributed over a particular variable. For example, in one of our use-case scenarios, the x-axis encodes temporal information (in years), while the y-axis encodes the counting of publications co-authored by an author or a set of authors. The data is displayed as a single bar per time period or multiple colored bars to represent categorical information of attributes. The **listing view** displays the elements that form the relationship between two or more nodes in the graph. Each item of the list is linked to a descriptive web page in the dataset where the user can obtain more information about it.

Fig. 3 shows an overview of the exploration process using MGExplorer. It starts with a query panel, where the user can choose a SPARQL endpoint and a predefined query (Fig. 3a), or directly with a graph view of the data (Fig. 3b), when the visualization is launched from the SPARQL query editor. From the graph view, the user can select nodes of interest to subset the data and explore it using other views, such as a temporal distribution (Fig. 3c) or a listing of items (Fig. 3d). The views are connected via line segments to represent their dependencies and enable retracing the exploration path. This same information can be retrieved through a history panel that is progressively completed with provenance information (Fig. 3e). To avoid clutter and help users focus on the relevant information to the ongoing analysis, users can hide any of the displayed views, which they may revisit later using the history panel. The input data (defined by the SPARQL query) is the reference data for selection operations throughout the whole exploration process. The system supports data and view selection, allowing users to specify subsets of interest from the whole input graph and suitable views to explore them. Upon selection of elements, the system filters the input dataset accordingly, and the resulting subset undergoes a transformation and mapping process that properly filter and reshape the data to be visualized with the selected visualization technique. The history records information about the selection operation, the data subset, the chosen view, and the transformed data.



SPARQL Endpoint * https://data.archives-ouvertes.fr/sp
Query * Query 4. Scenario 4: Co-occuren
Period From 2015 To 2021
Institution Laboratoire d'Informatique, Signa
Visualization technique * Graph view

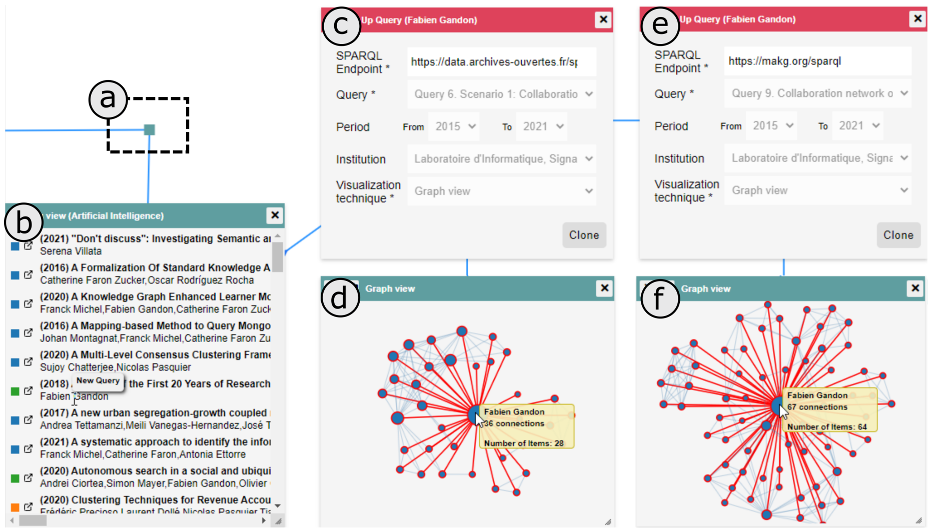
Initial Query
Graph view
Distribution view (Artificial Intelligence)
Listing view (Artificial Intelligence)

(2021) "Don't discuss": Investigating Semantic and Argumentative Features for Supervised Propagandist Message Detectio
Serena Villata
(2016) A Formalization Of Standard Knowledge And Skills For The French Elementary School Curricula
Catherine Faron Zucker,Oscar Rodríguez Rocha
(2020) A Knowledge Graph Enhanced Learner Model to Predict Outcomes to Questions in the Medical Field
Franck Michel,Fabien Gandon,Catherine Faron Zucker,Oscar Rodríguez Rocha,Antonia Ettorre
(2016) A Mapping-based Method to Query MongoDB Documents with SPARQL
Johan Montagnat,Franck Michel,Catherine Faron Zucker
(2020) A Multi-Level Consensus Clustering Framework for Customer Choice Modelling in Travel Industry
Sujoy Chatterjee,Nicolas Pasquier

Fig. 3. Overview of MGExplorer exploration process. In the query panel (a), the user chooses a SPARQL endpoint and a predefined query, which results are displayed in the graph view (b). By right-clicking on a node of interest, the user can subset the data and explore it through different views such as a distribution (c) or listing view (d) of items. The lines between views represent their dependency, which is also displayed in the history view (e).

2.5 Visualizing multiple SPARQL endpoints during exploration

The tool proposes a feature, called *Follow-up Queries*, that allows users to simultaneously explore multiple SPARQL result sets on the visualization dashboard [24]. This feature supports exploratory tasks such as (i) the comparison of a particular phenomenon described by different KGs and (ii) the inclusion of complementary data (from the same or a different SPARQL endpoint) to enrich the ongoing analysis. Fig. 4 illustrates the usage of follow-up queries subsequent to the exploration process depicted in Fig. 3. To avoid clutter, we first hide the views that are no longer necessary, which replaces the view by an icon (Fig. 4a) that serve to retrace the exploration process and is interactive to allow users to display the view again by clicking on it. In the listing view (Fig. 4b), suppose we are interested in the paper entitled "A Survey of the First 20 Years of Research on Semantic Web and Linked Data" and we want to know more about the author, "Fabien Gandon" and, particularly, about his scientific collaborations. For that purpose, we will query the HAL SPARQL endpoint to retrieve the coauthorship network of "Fabien Gandon". We right-click on the name of the author and select the option "New Query" (Fig. 4b), which instantiates a query panel giving the author's name as input data to be used as a parameter in the new query. As for the initial query, we select the endpoint of interest and the query (Fig. 4c), which results are displayed on a new graph view (Fig. 4d). This process allowed us to bring complementary data to the exploration process.



(2021) "Don't discuss": Investigating Semantic a
Serena Villata
(2016) A Formalization Of Standard Knowledge A
Catherine Faron Zucker,Oscar Rodríguez Rocha
(2020) A Knowledge Graph Enhanced Learner Mo
Franck Michel,Fabien Gandon,Catherine Faron Zuck
(2016) A Mapping-based Method to Query Mongo
Johan Montagnat,Franck Michel,Catherine Faron Zu
(2020) A Multi-Level Consensus Clustering Frame
Sujoy Chatterjee,Nicolas Pasquier
(2018) the First 20 Years of Research
Fabien Gandon
(2017) A new urban segregation-growth coupled
Andrea Tettamanzi,Meili Vanegas-Hernandez,José T
(2021) A systematic approach to identify the info
Franck Michel,Catherine Faron,Antonia Ettorre
(2020) Autonomous search in a social and ubiqui
Andrei Ciortea,Simon Mayer,Fabien Gandon,Olivier
(2020) Clustering Techniques for Revenue Accou
Frédéric Precioso,Laurent Dollé,Nicolas Pasquier,Ti

SPARQL Endpoint * https://data.archives-ouvertes.fr/sp
Query * Query 6. Scenario 1: Collaboratio
Period From 2015 To 2021
Institution Laboratoire d'Informatique, Signa
Visualization technique * Graph view

SPARQL Endpoint * https://makg.org/sparql
Query * Query 9. Collaboration network o
Period From 2015 To 2021
Institution Laboratoire d'Informatique, Signa
Visualization technique * Graph view

Fig. 4. Usage of follow up queries. We hide the views that are no longer necessary (a) and launch a new query by right-clicking on an item displayed on the listing view (b). In the new query panel (c), we choose an endpoint and query, which results are displayed in a new graph view (d). We can clone the query to reuse information, which creates a new query panel (e) where we can modify the information if necessary and relaunch the query. The results are then displayed in a new graph view (f).

Up to this point, we have explored the scientific collaboration network of "Fabien Gandon" from the perspective of the HAL knowledge graph. Now, let us compare these data with the coauthorship network of this researcher retrieved from another KG (i.e., the Microsoft Academic Knowledge Graph[4]. The tool allows us to clone the query view to reuse information and speed up the process. In the cloned query view (Fig. 4e), we change the SPARQL endpoint to MAKG and select query 9, the results of which can be seen in a new graph view (Fig. 4f). We can compare these visualizations side-by-side, where we can quickly observe that the network found in MAKG is slightly larger than that found in HAL. By hovering over the node that represents "Fabien Gandon" in both node-link diagrams, we observe that this author had 36 co-authors between 2015 and 2021 in 28 scholarly articles in the network retrieved from HAL. For the same period, the MAKG provided a network where this author had 64 co-authors through 64 scholarly articles.

2.6 Transformation Engine

The LDViz transformation engine consists of a converter module from SPARQL JSON results to the MGExplorer data model and a set of algorithms (i.e., mappers) that process subsets of data defined during the exploratory process via

[4] Available at https://makg.org/sparql

visual querying operations and map the resulting data to a particular visualization technique, also interactively chosen by the user.

From SPARQL Results to MGExplorer Data Model. The system receives the SPARQL JSON results set, which undergoes a transformation process to extract an attributed graph, encoded in the JSON format, that will serve as input data to MGExplorer. In addition to identifying mandatory and optional variables from the dataset, the process also derives indicators to describe the relationship between each pair of nodes, such as the total count of items and the count of items per type, when this information is provided.

MGExplorer Mappers. Every selection operation triggers a transformation process that filters and transforms the data and maps it to the selected visualization technique via: the *cluster view mapper*, which extracts clusters of nodes grouped according to the existing links among them, e.g., in a co-authorship network, the algorithm detects groups of authors co-authoring the same publication(s); the *egocentric view mapper*, which extracts pairwise relationships between the selected node and the other nodes in the subset; the *pairwise relationship view mapper*, which extracts pairwise relationships by analyzing every possible combination of pairs of nodes within the subset; the *distribution view mapper*, which extracts the distribution of items in the subset according to a particular attribute (e.g., date); and the *listing view mapper*, which extracts the list of links in the graph and their descriptive information (if provided). Regardless of the resulting relationship type, every mapper keeps information on the count and type of items per relationship.

3 KG Exploration Methods and SPARQL Queries

LDViz covers three domains of data exploration: (i) RDF graph/vocabulary inspection, (ii) RDF graph summarization, and (iii) exploratory search. In this section, we present the spectrum of SPARQL queries capable of extracting the necessary data from RDF graphs to support such as data. We then demonstrate the generic use of LDViz by applying those queries on two distinct SPARQL endpoints giving access to the DBpedia FR dataset[5] (over 400 million triples describing the content generated in the Wikipedia project) and the *HAL* dataset[6] (an open archive for scientific publications in all domains).

3.1 RDF Graph/Vocabulary Inspection

When working with the Semantic Web, a recurring task is to inspect the RDF graph and its ontology to learn its content. In particular, we consider exploration tasks where the user wants to (1) display the RDF graph with no particular goal in mind and (2) get an idea of the ontology used in the RDF graph. The RDF graph can be extracted through a simple SPARQL SELECT query retrieving

[5] SPARQL endpoint: http://fr.dbpedia.org/sparql

[6] SPARQL endpoint: http://sparql.archives-ouvertes.fr/sparql.

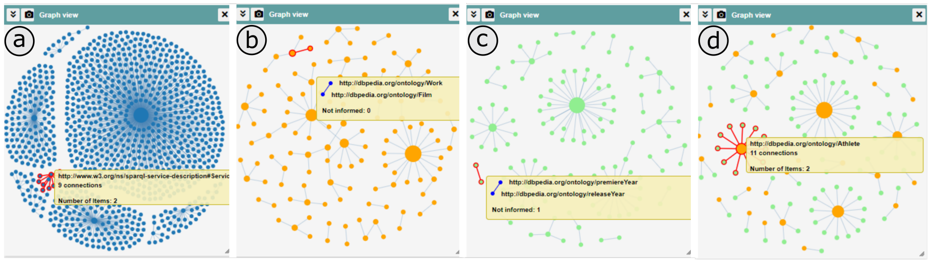

Graph view
a
http://www.w3.org/ns/sparql-service-description#Servic
9 connections
Number of Items: 2
Graph view
b
http://dbpedia.org/ontology/Work
http://dbpedia.org/ontology/Film
Not informed: 0
Graph view
c
http://dbpedia.org/ontology/premiereYear
http://dbpedia.org/ontology/releaseYear
Not informed: 1
Graph view
d
http://dbpedia.org/ontology/Athlete
11 connections
Number of Items: 2

Fig. 5. Graph view of an extract of DBpedia's (a) RDF graph, (b) hierarchy of classes, (c) hierarchy of properties, and (d) signatures of properties linking classes (orange) and properties (light green).

every triple `?s`, `?p`, `?o` in the graph, without specific matching (Listing 1.3a). To support the exploration of the RDF vocabularies, we define three SPARQL query templates based on the RDF Schema data-modeling vocabulary to retrieve the (a) *hierarchy of classes*, defined by the `rdfs:subClassOf` property (Listing 1.3b), (b) the *hierarchy of properties*, defined by the `rdfs:subPropertyOf` property Listing 1.3c), and (b) the *signature of properties*, defined by the properties `rdfs:domain` and `rdfs:range`, which give the class to which the subject of an RDF statement using a given property belongs, and the class of its object (value), respectively (see Listing 1.3d). These SPARQL query templates are generic enough to retrieve information from any SPARQL endpoint, as long as it includes the RDF Schema description.

To demonstrate the feasibility of these SPARQL queries, we apply them to the DBPedia endpoint and visualize the results using LDViz. Fig. 5a shows an interactive graph view of the 1000 first statements in the DBpedia graph. The graph views in Figures 5b-d show the above mentioned methods of RDF vocabulary inspection: hierarchy of classes and properties, and the signatures of properties. Users can hover over nodes to inspect and navigate within hierarchies and explore property signatures by hovering over nodes that represent properties to inspect their signature or classes to identify all the properties to whose signatures the selected class belongs (e.g., `dbo:Athlete` is related to eleven properties). In this example, we leverage the GSS feature to assign meaningful visual elements to certain variables as shown in Fig 5d, where color encodes property (light green) and class nodes (orange), assisting visual search and understanding of relationships between nodes of different types.

```
(a) select * where { ?s ?p ?o }         (b) select * where { ?s ?p ?o
                                            filter(?p = rdfs:subClassOf) }
(c) select * where { ?s ?p ?o           (d) select * where { ?s ?p ?o
    filter(?p = rdfs:subPropertyOf) }       filter(?p = rdfs:domain ||
                                                   ?p = rdfs:range)}
```

Listing 1.3. SPARQL query templates for RDF graph/vocabulary inspection via the (a) RDF graph, (b) hierarchy of classes, (c) hierarchy of properties, and (d) signature of properties.

3.2 RDF Graph Summarizations

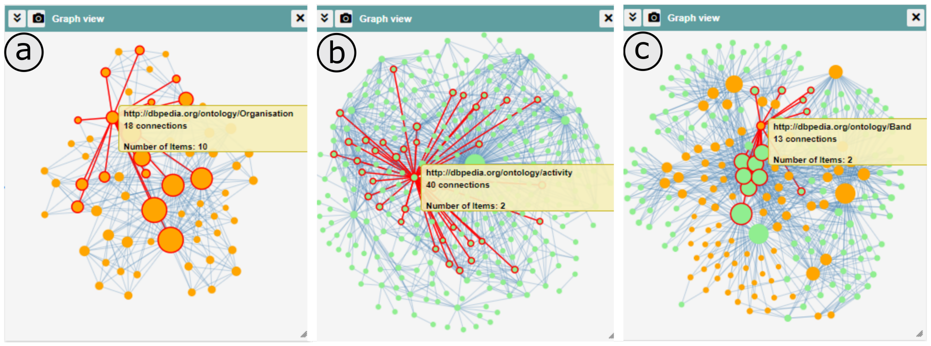


Fig. 6. Graph views of DBpedia RDF summarizations representing (a) class paths, (b) property paths, and (c) paths of type class → property → class.

A benefit of visualization for exploring RDF graphs relies on its capacity to reveal tendencies and patterns within the data. However, visualization knows its limitations as one tries to display millions of triples on the screen, resulting in a huge and cluttered graph that hinders the discovery of meaningful information. Structural RDF graph summarization addresses this issue by providing indices or summaries of RDF graphs to aggregate the triples in meaningful ways. We consider three methods of RDF graph summarization, which we support through three SPARQL query templates (Listing 1.4) capable of extracting (i) the existing paths between classes of resources in an RDF graph, (ii) the existing property paths between the resources of the graph, or (iii) the paths between classes and properties (i.e. Class → Property → Class path). To demonstrate the feasibility of these queries, we applied them on the DBPedia SPARQL endpoint. The graph view in Fig. 6a summarizes the DBPedia RDF graph by showing how classes are connected through properties, while the graph view in Fig. 6b shows how properties are connected through resources. Finally, Fig. 6c shows how properties and classes are connected together through resources.

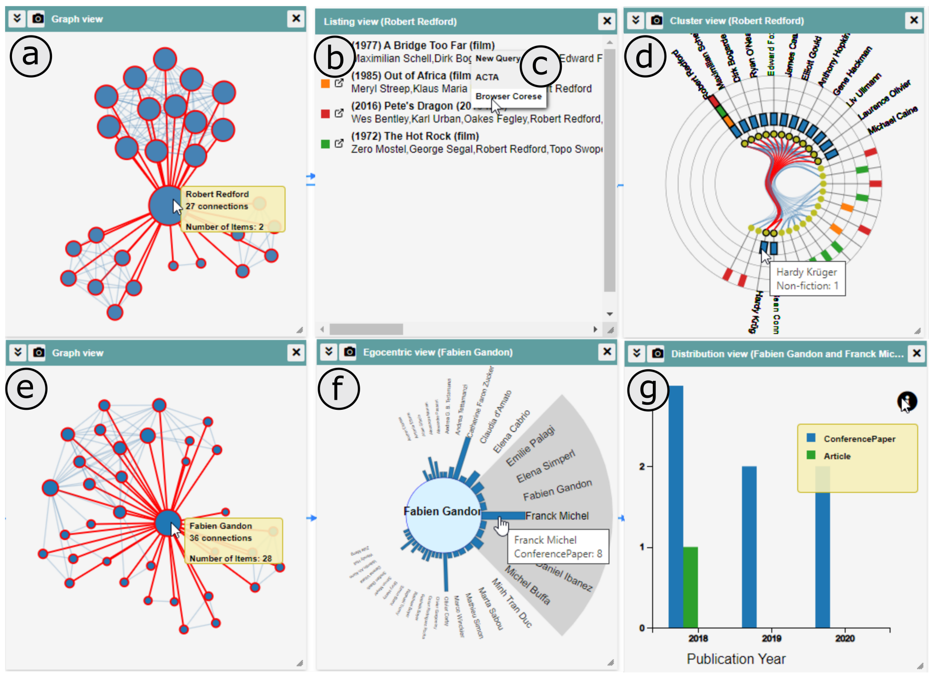
```
prefix ldv: <http://ldv.fr/path/>
(a) select distinct ?s ?p ?o
    where { ?a ?p ?b . ?a a ?s . ?b a ?o }

(b) select distinct ?s (ldv: as ?p) ?o where {
    ?x ?s ?y . ?y ?o ?z . filter (?s != ?o)}

(c) select distinct ?s (ldv: as ?p) ?o where {
    {?a ?b ?c. ?a a ?s . bind (?b as ?o)} UNION
    {?a ?b ?c. ?c a ?s . bind (?b as ?o)}}
```

Listing 1.4. SPARQL query templates for exploring RDF summarizations through (a) class paths, (b) property paths, and (c) paths of type class → property → class.

3.3 Exploratory Search of Knowledge Graphs



(1977) A Bridge Too Far (film)
Maximilian Schell,Dirk Bog ... Edward F

(1985) Out of Africa (film
Meryl Streep,Klaus Maria ... t Redford

(2016) Pete's Dragon (20
Wes Bentley,Karl Urban,Oakes Fegley,Robert Redford,

(1972) The Hot Rock (film)
Zero Mostel,George Segal,Robert Redford,Topo Swope

Fig. 7. Exploratory path of Robert Redford's co-starring network (a-d) and Fabien Gandon's co-authorship network (e-g).

The set of KG exploration methods presented above are useful to support data producers while inspecting or discovering the RDF graph. As for any dataset, KGs provide data that describes a particular phenomenon, which analysis could support decision-making processes on a particular application domain. Therefore, we support exploratory search in KG starting from a question or hypothesis, which is then formulated as SPARQL query to retrieve an initial dataset used in the exploration. Hereafter, we define SPARQL query templates to support exploratory search in KGs with focus on relationship networks.

The SPARQL query template presented in Listing 1.5a is able to retrieve the co-starring network of a given artist described by the DBPedia KG. In this example, we focus on Robert Redford, which resulting graph view shown in Fig. 7a consists of 28 nodes (actors) and 137 links (movies). We see that Redford has 27 co-stars across four movies where details are available at the Listing view (Fig. 7b). The resources can be explored using the Corese browser (Fig. 7c) or any other service enabled in the GSS. Furthermore,the cluster view (Fig. 7d) shows Redford's co-stars grouped by movie.

```
(a) prefix dbo: <http://dbpedia.org/ontology/>
    prefix dbp: <http://dbpedia.org/property/>
    select * where { ?x rdfs:label {artist name} .
        ?p dbo:starring ?x, ?a1, ?a2; rdfs:label ?label;
            dbp:released ?date ; dbp:genre ?type .
        ?a1 rdfs:label ?s . ?a2 rdfs:label ?o . }

(b) prefix dc:<http://purl.org/dc/terms/>
    prefix foaf:<http://xmlns.com/foaf/0.1/>
    prefix hsc:<http://data.archives-ouvertes.fr/schema/>
    select * where { ?p dc:creator ?x, ?x1, ?x2 ;
        dc:type ?type ; dc:title ?label ; dc:issued ?date.
        ?x hsc:person ?a . ?a foaf:name {researcher name}.
        ?x1 hsc:person ?a1 . ?a1 foaf:name ?s .
        ?x2 hsc:person ?a2 . ?a2 foaf:name ?o . }
```

Listing 1.5. SPARQL query template for retrieving (a) the co-starring network of a particular artist from DBpedia and (b) the co-authorship network of a particular researcher from HAL

The SPARQL query template in Listing 1.5b allows to retrieve the co-authorship network of any researcher from the HAL SPARQL endpoint. In this example, we focus on the co-authorship network of Fabien Gandon between the year of 2015 and 2021. The resulting graph view in Fig. 7e is formed by 35 nodes (authors) and 109 links (publications). We observe that Fabien Gandon has 36 co-authors via 28 publications. We further explore the pairwise relationship between this researcher and his peers using the pairwise relationship view, where we can identify the researcher with whom he has the most publications. As an example, we focus on the co-authorship between him and Franck Michel, which resulted in 8 scholarly articles during that period (Fig. 7f). Further, we explore these articles over time using a distribution view (Fig. 7e), where we can observe a constant collaboration with the most articles being published together in 2019. The distribution view also displays the publications' types (i.e., conference paper, article), showing that they have mostly published conference papers together.

4 Coverage Analysis

To demonstrate the extent to which LDViz can support the exploration of LD datasets, we implemented a script that tested 419 different SPARQL endpoints to identify whether the SPARQL result set could be visualized by our tool.

4.1 Data

The 419 SPARQL endpoints used in this analysis were obtained from IndeGx [21], a framework designed to index public KGs that are available online through a SPARQL endpoint. The indexing process uses SPARQL queries to either extract the available metadata from a KG or to generate as much metadata as the endpoint allows it. The generated metadata not only describes KGs and their endpoints but also conveys an estimation of certain quality criteria. The queries used by IndeGx to index KGs and their endpoints are available in a public repository at https://github.com/Wimmics/dekalog, and the results of its indexations are publicly available through a SPARQL endpoint at http://prod-dekalog.inria.fr/sparql, from which we retrieved the list of endpoints using the query presented in Listing 1.6. These SPARQL endpoints are present in the dataset generated by IndeGx because they appeared in different publicly available catalogs of datasets. In particular, they were retrieved from the LOD Cloud website[7], Yummy Data [34], Wikidata[8], Linked Wiki[9], SPARQLES [33] and the OpenLink company endpoint[10].

```
prefix index: <http://ns.inria.fr/kg/index\#>
prefix desc: <http://www.w3.org/ns/sparql-service-description\#>
SELECT DISTINCT ?endpointUrl where {
    GRAPH ?g {  ?metadata index:curated ?dataset .
        ?dataset desc:endpoint ?endpointUrl .  }  }
```

Listing 1.6. SPARQL query used to retrieve the list of available endpoints from the IndeGx RDF graph.

4.2 Procedure

The queries in the exploratory search category require a knowledge of the RDF graph and vocabulary to retrieve suitable data to start the exploration. Thus, we ran the evaluation using only queries that serve to inspect the RDF graph or vocabulary, and those that provide RDF summarizations as they are rather generic to any endpoint. The only specific vocabulary used by these queries is the RDF Schema, which provides a data modeling vocabulary for RDF data and would be therefore expected to appear in most RDF graphs. We implemented a *nodejs* script that applies each query against every one of the 419 SPARQL endpoints retrieved from the IndeGx endpoint using the `fetch` API provided by the `node-fetch` module. We limited each query to 10 solutions to speed up the process, as our goal was to inspect the resulting data format to check whether we could visualize it using LDViz; the actual data was not important for this analysis. A request would have mainly two possible outcomes. In case of a successful request, we inspect the resulting data format to verify whether it matches the SPARQL JSON result set defined by W3C Recommendation. If the data does not match the expected format, we inspect it further to identify its format, which may sometimes be HTML or CSV, for instance. In case of a failed request, we inspect the error thrown to understand why we were unable to retrieve data from that particular endpoint.

4.3 Results

Figure 8 presents a TreeMap graph showing the distribution of different responses obtained while querying 419 SPARQL endpoints obtained from IndeGx [21]. The top level of the TreeMap graph contains seven rectangles each of which covers a

[7] https://lod-cloud.net
[8] https://www.wikidata.org/
[9] https://linkedwiki.com/
[10] http://lod.openlinksw.com/sparql/

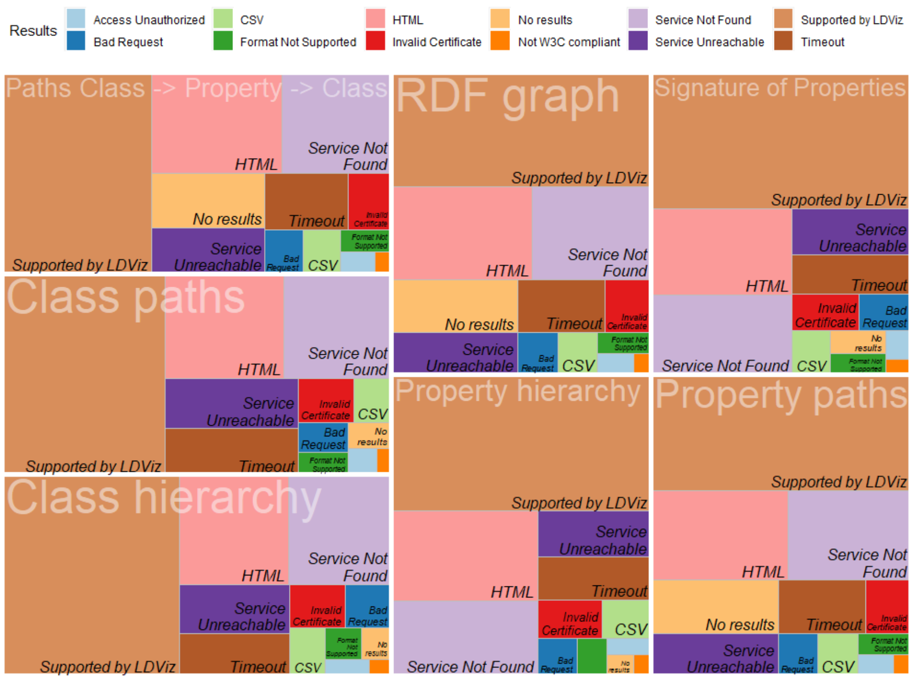


Fig. 8. Summary of results per type of query. In the case where results cannot be visualized with LDViz, we display the issues encountered while querying the SPARQL endpoints.

specific query type (i.e Paths→Properties→Class, Class paths, Class hierarchy, RDF graph, property hierarchy, signature of properties, and properties path). These rectangles are further divided in smaller colored rectangles that summarize the results obtained per query type including SPARQL endpoints supported by LDViz and issues encountered while accessing the endpoints (e.g., Access Unauthorized, Service not found, etc.). The size of the rectangle encodes the number of results obtained for each query.

On average, 41.77% of the SPARQL endpoints returned a valid result set that could be explored using LDViz. We noticed that the queries seeking for class and property hierarchy, and signature of properties were slightly less successful than the remaining, where only about 38.19% of SPARQL endpoints returned a valid result set. Regarding the issues found while querying the SPARQL endpoints, we could identify 11 different reasons for why it cannot be explored using LDViz. Table 4 summarizes the percentage of issues per query type. Hereafter we present the issues in decreasing order of occurrence:

– **HTML**: About 16.06% of the requests returned an HTML object, which may contain valid results from the SPARQL endpoint, but cannot be processed by the LDViz transformation engine.

- **Service Not Found**: The SPARQL endpoint could not be found (request status 404 and 410). We encountered this issue in about 14.18% of requests.
- **Service Unreachable**: This issue is identified when the connection is refused by the server (throwing the ECONNREFUSED error), or the protocol encountered an unrecoverable error for that endpoint (throwing the EPROTO error). On average, 7.06% of requests encountered this issue. We noticed that this issue appeared slightly more often for the SPARQL query recovering the RDF summarization through class paths then for the remaining, where we observed the issue in 8.59% of requests.
- **Timeout**: About 6.27% of the requests encountered a timeout issue. This is due to the request response not being received within the default timeout of the fetch request, which is of about 300 seconds (request statuses 408 and 504) or the Virtuoso server estimating the query processing time to be longer than its established timeout of 400 seconds.
- **No results**: This issue means that the request returned a valid JSON object, but the `bindings` array was empty. In average, 4.30% of SPARQL endpoints did not provide results to our queries. However, once again, we observe that this number is higher for SPARQL queries seeking for the signature of properties, class, and property hierarchies, where we observe that about 8.35% of endpoints did not provide results against an average of only 1.25% of endpoints not providing results for the remaining queries.
- **Invalid Certificate**: The request could not be completed due to an invalid certificate on the SPARQL endpoint side. This issue was observed in about 3.10% of requests, which correspond to 13 SPARQL endpoints.
- **CSV**: On average, 2.18% of the requests returned a string object which content follows a CSV format. The result set may contain valid data but cannot be processed by the LDViz transformation engine.
- **Bad Request**: The request could not be fulfilled due to bad syntax (request status 400). This error was thrown by 2.18% of requests, which correspond to 9 to 11 SPARQL endpoints. We could observe that the SPARQL queries seeking for the RDF graph and an RDF graph summarization through class paths were slightly less affected than the remaining.
- **Format Not Supported**: Requests for 6 different SPARQL endpoints have responded with this error (1.43% of requests), which means that the server can only generate a response that is not accepted by the client (status 406).
- **Access Unauthorized**: This issue encompasses the following request responses: the server refuses to respond (status 401 and 403), and authentication is required (status 407 and 511). We observed that three endpoints (0.95% of requests – 4 SPARQL endpoints) required authentication, which we could not provide.
- **Not W3C compliant**: The request responded with a JSON object that does not follow the JSON format specified by the W3C Recommendation. This issue was observed in 2 SPARQL endpoints (0.48%).

To better understand the issues, we further inspected some of the SPARQL endpoints using the KartoGraphi application[11] [21], which provides an overview of the state of the 419 endpoints used in this analysis through the metadata generated by IndeGx. It shows, for instance, that not every SPARQL endpoint is transparent regarding the used vocabularies (only 110 endpoints provide the list of vocabularies used). Furthermore, only 95 endpoints contain a RDF Schema vocabulary, revealing that not every SPARQL endpoint contains the description of RDF Schema, which may explain the higher rate of empty results for the queries retrieving the hierarchies and signature of properties using RDF Schema properties such as `subClassOf`, `subPropertyOf`, `domain`, and `range`. Moreover, we noticed that some of the SPARQL endpoints do not support the majority of the SPARQL features, which could explain why they did not recognize the syntax of the queries, throwing a bad request error.

5 Related Work

A complete survey of tools designed for LOD exploration is beyond the scope of this paper. For that, we suggest the reading of the comprehensible survey of 70 such tools [12], previous surveys of linked-data based exploration systems [22], and the definitions and models of exploratory search [26]. In this section, we focus on LOD visualization tools that support the exploration of (i) OWL or RDF Schema, (ii) the RDF graph, and (iii) custom datasets represented according to data types, while examining their support to perform tasks of (i) RDF graph/vocabulary inspection, (ii) RDF summarization, and (iii) exploratory search. Table 2 summarizes the reviewed tools according to supported data format, access methods, represented aspects of data, visualization, and interaction tools.

OWL/RDF Schema Visualization. Kremen et al. [20] represent the structure of RDF datasets and the relationship with other datasets by using class/properties statistics, spatial and temporal information, and a dataset summary. Similarly, the tool proposed by Anutariya and Dangol [2] uses a node-link diagram to visualize schema information inferred via SPARQL queries using ontological characteristics of the triples in the LOD data sources.

RDF Graph Visualization. Aiming at simplifying the exploration of large RDF graphs, various visualization tools have been proposed in the literature to support the progressive visual exploration of LOD, which would simply require a particular resource or a RDF dataset as starting point [10, 11, 19]. In such tools, the RDF graph is represented via a node-link diagram and the user can incrementally reveal or hide neighboring resources via selection operations to explore and visualize relevant data from very large RDF graphs [11], while discovering linked RDF graphs in the Web [19], and inspecting information and internal relations of data subsets [10]. To assist the user in interpreting all nodes and links of an RDF graph as knowledge structures by keeping only interesting triples, Chawuthai and Takeda [8] use graph simplification methods to visualize

[11] Accessible at http://prod-dekalog.inria.fr/

Ref.	Year	Tool	Input Data	Rep.	Data Access	Visualization	Interaction
[13]	2020	S-Paths	RDF Dataset	Per Datatype	RDF Dump	CO, G, H, P, PT	CC, F
[14]	2018	N/A	RDF Dataset	RDF Graph	RDF Dump	H, PT, R	CC, F, M
[2]	2018	VizLOD	RDF Dataset	OWL/RDF Schema	SPARQL / RDF Dump	R	F, V
[20]	2018	Dataset Dash-board	RDF Dataset	OWL/RDF Schema	RDF Dump	R, TT	F, V
[19]	2018	LOD Explorer	RDF Dataset	RDF Graph	JSONP	R	DD, F
[18]	2018	JLO/GIG	SPARQL Result Sets	RDF Graph	SPARQL	CL, R	CC, F, M, V
[8]	2016	N/A	RDF Dataset	RDF Graph	SPARQL construct	R	CC, F
[27]	2016	N/A	SPARQL Result Sets	Per Datatype	SPARQL / RDF Dump	CO, G, P, R	N/A
[32]	2015	LinkDaViz	RDF Dataset	Per Datatype	RDF Dump	CO, D, G, P, T	CC, E, M, V
[10]	2014	LOD/ VizSuite	RDF Dataset	RDF Graph	SPARQL	R	CC, F
[17]	2013	VisualBox	SPARQL Result Sets	Per Datatype	SPARQL	G, R, T	E, F
[3]	2012	LDVM	Non/RDF Dataset	Per Datatype	RDF Dump	G, H, P	F, M, V
[30]	2012	Sgvizler	SPARQL Result Sets	Per Datatype	SPARQL select	CO, D, G, H, R, P, T	N/A
[35]	2011	ViziQuer	SPARQL Result Sets	OWL/RDF Schema	SPARQL	R	F, V
[11]	2007	PGV	RDF Dataset	RDF Graph	SPARQL	R	CC, F, V

Table 2. Summary of related work: publication reference, year, name (if provided), input data type, represented information, data access, visualization type (**CO**: comparison, **CL**: clustering, **D**: distribution, **G**: geographical, **H**: hierarchical, **P**: proportional, **PT**: patterns, **R**: relationship, **T**: temporal, **TT**: text and table), and interaction operations (**CC**: chart customization, **E**: chart export, **F**: data filtering, **M**: visual mapping, **V**: view operations, **DD**: details on demand). **N/A** stands for non-available. Source: Menin et al. [25]

an RDF graph, which remove redundant triples to present a sparse graph to the user, while ranking triples according to topics of interest.

Frasincar et al. [14] propose an RDF data format plugin for a general-purpose visual environment that supports browsing and editing graph data. Users can define new operations for data processing, visualization, and interaction while being able to modify visual mapping by changing the shape, size, and color of nodes and edges. Graziosi et al. [18] provide a user-friendly SPARQL query builder to support non-programmers users in extracting data from the Web and exploring it through a node-link diagram. Likewise, users can modify visual attributes of nodes (shape, color, border, etc.) via a customizable template for the visualization of entities and properties.

Visualization per Datatype. In an attempt to improve the visualization of LOD by considering the characteristics of the data, a few tools have been proposed to analyze the RDF vocabulary of the input data to visualize it accordingly (e.g., data containing properties such as `xsd:date` and `ical:dtstart` would be visualized through timeline or calendar visualizations) [3,27,32]. Similarly, the S-Paths visualization tool [13] supports the visualization of resources sets based on semantic paths by identifying and ranking a set of visualization techniques suitable to explore the data. Via interaction tools, the user can explore different resource sets and/or use different visualization techniques to get a different perspective to the dataset delivered via different semantic paths.

The Visualbox tool [17] generates graph, temporal, and geographical visualizations to explore SPARQL result datasets; it also exports the visualization in a format suitable for incorporation into hypertextual documents. Similarly, the JavaScript wrapper proposed by Skjaeveland [30] generates visualizations of SPARQL result sets via HTML elements, such as web components, embedded with SPARQL SELECT queries, which are rendered to contain the specified visualization type on page load or function call.

Tool & Ref.	**RDF graph / vocabulary inspection**	**RDF Summarization**	**Exploratory Search**
S-Paths [13]			✓
[14]	✓		
VizLOD [2]	✓		
Dataset Dashboard [20]	✓	✓	
LOD Explorer [19]	✓		
JLO/GIG [18]	✓		✓
[8]	✓	✓	
[27]	✓		✓
LinkDaViz [32]	✓		
LOD/ VizSuite [10]			✓
VisualBox [17]			✓
LDVM [3]			✓
Sgvizler [30]			✓
ViziQuer [35]	✓		
PGV [11]	✓		
LDViz	✓	✓	✓

Table 3. Summary of related work regarding task support: RDF profiling, RDF summarization, or exploratory search.

Table 3 presents these related works according to the type of KG exploration they support, i.e. RDF graph/vocabulary inspection, summarization, and exploratory search. To our knowledge, there is no LOD visualization tool that supports all three types of analysis, which can be achieved with LDViz. In particular, the advantage of our approach compared to existing solutions relies on a flexibility that allows users to define meaningful datasets via SPARQL SELECT queries applied to any SPARQL endpoint, so that they can explore multiple aspects of RDF datasets, as well as to progressively explore the LOD Cloud through the usage of follow-up queries launched on the fly to include external data into the exploration process. It also allows users to perform exploratory searches using various complementary visualization techniques, instantiated on demand, focusing on meaningful subsets of data according to the task at hand, instead of a single visualization technique that represents the whole data set, restricting the analysis to a single view of the data.

6 Discussion, Conclusion and Future Work

In this paper, we present a web-based interactive visualization tool for LOD exploration called LDViz. It provides access to any SPARQL endpoint by allowing users to perform searches with SPARQL queries and visualize the results via multiple perspectives delivered through complementary visualization techniques.

KG exploration methods. Our approach supports the exploration of KG through a set of methods which we support via a set of SPARQL query templates that allow (i) RDF graph/vocabulary inspection, (ii) RDF summarizations exploration, and (iii) exploratory search. We defined the scope of SPARQL queries through templates that can be reused over any SPARQL endpoint, either directly or after slight modifications to accommodate the RDF vocabulary. We demonstrated their usage and feasibility through a set of use case scenarios and a coverage analysis that apply those queries over 400 SPARQL endpoints.

Visual design and interactions. We support exploration search via MG-Explorer, a visualization tool for progressively exploring multidimensional network data via multiple complementary views. Users can select subsets of data through visual queries and display the results in a separate view that shows a different perspective to the data. The multiple views can be hidden, revisited, and arranged in the display area in meaningful ways to support efficient data exploration while reducing cognitive overhead and clutter-related issues. The tool provides yet a *follow up query* feature that allow the user to bring external data into the exploration process via predefined queries processed on-the-fly. The different datasets can be simultaneously explored in the same dashboard, enriching the ongoing analysis, while allowing the progressive exploration of the Web.

User support. When exploring KGs, a great deal of time and effort is spent in testing and debugging SPARQL queries to ensure that the resulting data is sufficient to accomplish the task at hand. Thus, we support data producers and analysts via a SPARQL query editor, where users can test and debug their queries, or import predefined queries, which they may use as templates to create new queries, simplifying the process. Furthermore, to support domain users on their decision-making processes without having to deal with the complexity of the SPARQL language, LDViz includes an interface where users can perform exploratory search through predefined queries. The tool is available at http://dataviz.i3s.unice.fr/ldviz.

Generalization. Through the scope of SPARQL queries defined in this paper, our results showed that LDViz can support the exploration of KGs from about 42% of the 419 analyzed SPARQL endpoints. We noticed that certain queries, such as the ones describing the signature of properties, class and property hierarchies of KGs were less successful encountering issues such as bad request and no results more often than the remaining queries, which may be explained by the SPARQL endpoint missing RDF Schema vocabulary description. In general, we observe that most issues encountered were rather caused by accessibility limitations at the SPARQL endpoint side. We follow the W3C standards, as we believe this ensures the accessibility and homogeneity of data throughout the Web. However, this could be considered a limitation of our approach, as it prevents the visualization of SPARQL endpoints that are not W3C compliant (about 18% of endpoints in our analysis).

Usability and Suitability. Although our use case scenarios and our coverage analysis are enough to support the feasibility and genericity of our approach, user-based evaluations are essential and should be performed to determine the usability and suitability of LDViz. Thus, future work includes developing user-based evaluations to investigate the usability of LDViz to assist the resolution of these and other use cases by expert users in Semantic Web, as well as to assist decision-making processes via exploratory search of RDF graphs, involving expert users in diverse application domains.

Acknowledgements

We are grateful to Ricardo A. Cava, who provided us with the first version of MGExplorer, which was developed as part of his Ph.D. thesis at the Federal University of Rio Grande do Sul. C.D.S. Freitas is funded by the Brazilian funding agencies CNPq and CAPES (Finance Code 001). This work is also partially funded by University of Côte d'Azur through its IDEX$^{\mathrm{JEDI}}$ program (CC: C870A06232 EOTP: LINKED_OPEN_DATA DF: D103).

References

1. Antoniazzi, F., Viola, F.: RDF graph visualization tools: A survey. In: 2018 23rd Conference of Open Innovations Association (FRUCT). pp. 25–36. IEEE (2018). https://doi.org/10.23919/FRUCT.2018.8588069
2. Anutariya, C., Dangol, R.: VizLOD: Schema extraction and visualization of linked open data. In: 2018 15th International Joint Conference on Computer Science and Software Engineering (JCSSE). pp. 1–6. IEEE (2018). https://doi.org/10.1109/JCSSE.2018.8457325

3. Brunetti, J.M., Auer, S., García, R., Klímek, J., Nečaskỳ, M.: Formal linked data visualization model. In: Proceedings of International Conference on Information Integration and Web-based Applications & Services. pp. 309–318 (2013). https://doi.org/10.1145/2539150.2539162
4. Card, S.K., Mackinlay, J.D., Shneiderman, B. (eds.): Readings in Information Visualization: Using Vision to Think. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1999)
5. Cava, R., Freitas, C.D.S.: Glyphs in matrix representation of graphs for displaying soccer games results. In: The 1st Workshop on Sports Data Visualization. IEEE. vol. 13, p. 15 (2013)
6. Cava, R., Freitas, C.M.D.S., Winckler, M.: Clustervis: visualizing nodes attributes in multivariate graphs. In: Proceedings of the Symposium on Applied Computing. pp. 174–179 (2017). https://doi.org/10.1145/3019612.3019684
7. Cava, R., Freitas, C.M., Barboni, E., Palanque, P., Winckler, M.: Inside-in search: an alternative for performing ancillary search tasks on the web. In: 2014 9th Latin American Web Congress. pp. 91–99. IEEE (2014). https://doi.org/10.1109/LAWeb.2014.21
8. Chawuthai, R., Takeda, H.: Rdf graph visualization by interpreting linked data as knowledge. In: Joint International Semantic Technology Conference. pp. 23–39. Springer (2015). https://doi.org/10.1007/978-3-319-31676-5_2
9. Corby, O., Gaignard, A., Faron-Zucker, C., Montagnat, J.: KGRAM Versatile Data Graphs Querying and Inference Engine. In: Proc. IEEE/WIC/ACM International Conference on Web Intelligence. Macau (Dec 2012)
10. De Vocht, L., Dimou, A., Breuer, J., Van Compernolle, M., Verborgh, R., Mannens, E., Mechant, P., Van de Walle, R.: A visual exploration workflow as enabler for the exploitation of linked open data. In: IESD'14 Proceedings of the 3rd International Conference on Intelligent Exploration of Semantic Data. vol. 1279, pp. 30–41. CEUR-WS. org (2015)
11. Deligiannidis, L., Kochut, K.J., Sheth, A.P.: Rdf data exploration and visualization. In: Proceedings of the ACM first workshop on CyberInfrastructure: information management in eScience. pp. 39–46 (2007). https://doi.org/10.1145/1317353.1317362
12. Desimoni, F., Po, L.: Empirical evaluation of Linked Data visualization tools. Future Generation Computer Systems **112**, 258–282 (2020). https://doi.org/https://doi.org/10.1016/j.future.2020.05.038
13. Destandau, M., Appert, C., Pietriga, E.: S-Paths: Set-based visual exploration of linked data driven by semantic paths. Semantic Web **12**(1), 99–116 (2021). https://doi.org/10.3233/SW-200383
14. Frasincar, F., Telea, A., Houben, G.J.: Adapting graph visualization techniques for the visualization of rdf data. In: Visualizing the semantic web, pp. 154–171. Springer (2006). https://doi.org/10.1007/1-84628-290-X_9
15. Gandon, F.: A Survey of the First 20 Years of Research on Semantic Web and Linked Data. Revue des Sciences et Technologies de l'Information (Dec 2018). https://doi.org/10.3166/ISI.23.3-4.11-56
16. Gandon, F., Hall, W.: A Never-Ending Project for Humanity Called "the Web". In: WWW 2022 - ACM Web Conference. Lyon (virtual), France (Apr 2022). https://doi.org/10.1145/3485447.3514195
17. Graves, A.: Creation of visualizations based on linked data. In: Proceedings of the 3rd International Conference on Web Intelligence, Mining and Semantics. pp. 1–12 (2013). https://doi.org/10.1145/2479787.2479828

18. Graziosi, A., Di Iorio, A., Poggi, F., Peroni, S., Bonini, L.: Customising LOD views: a declarative approach. In: Proceedings of the 33rd Annual ACM Symposium on Applied Computing. pp. 2185–2192 (2018). https://doi.org/10.1145/3167132.3167367
19. Jacksi, K., Zeebaree, S.R., Dimililer, N.: LOD Explorer: Presenting the Web of Data. Int. J. Adv. Comput. Sci. Appl. IJACSA **9**(1) (2018)
20. Kremen, P., Saeeda, L., Blasko, M.: Dataset dashboard-a sparql endpoint explorer. In: VOILA@ ISWC. pp. 70–77 (2018)
21. Maillot, P., Corby, O., Faron, C., Gandon, F., Michel, F.: KartoGraphI: Drawing A Map Of Linked Data. In: The Semantic Web: ESWC 2022 Satellite Events: ESWC 2022 Satellite Events, Heraklion, Crete, Greece, May 29 – June 2, 2022, Revised Selected Papers. Springer-Verlag, Berlin, Heidelberg (2022)
22. Marie, N., Gandon, F.: Survey of linked data based exploration systems. In: IESD 2014 - Intelligent Exploitation of Semantic Data. Riva Del Garda, Italy (Oct 2014), https://hal.inria.fr/hal-01057035
23. Menin, A., Cava, R., Freitas, C.M.D.S., Corby, O., Winckler, M.: Towards a visual approach for representing analytical provenance in exploration processes. In: 2021 25th International Conference Information Visualisation (IV). pp. 21–28 (2021). https://doi.org/10.1109/IV53921.2021.00014
24. Menin, A., Do, M.N., Dal Sasso Freitas, C., Corby, O., Faron Zucker, C., Giboin, A., Winckler, M.: Using Chained Views and Follow-up Queries to Assist the Visual Exploration of the Web of Big Linked Data. International Journal of Human-Computer Interaction (2022), https://hal.archives-ouvertes.fr/hal-03518845
25. Menin, A., Faron, C., Corby, O., Freitas, C., Gandon, F., Winckler, M.: From Linked Data Querying to Visual Search: Towards a Visualization Pipeline for LOD Exploration. In: Proceedings of the 17th International Conference on Web Information Systems and Technologies - WEBIST, ISBN 978-989-758-536-4; ISSN 2184-3252. pp. 53 – 64 (Oct 2021). https://doi.org/10.5220/0010654600003058
26. Palagi, E., Gandon, F., Troncy, R., Giboin, A.: A Survey of Definitions and Models of Exploratory Search. In: ESIDA '17 - ACM Workshop on Exploratory Search and Interactive Data Analytics. pp. 3–8. Limassol, Cyprus (Mar 2017). https://doi.org/10.1145/3038462.3038465
27. Peña, O., Aguilera, U., López-de Ipiña, D.: Exploring lod through metadata extraction and data-driven visualizations. Program (2016)
28. Pietriga, E.: Semantic web data visualization with graph style sheets. In: Proceedings of the 2006 ACM symposium on Software visualization. pp. 177–178 (2006)
29. Recommentation, W.: Sparql 1.1 query results json format. https://www.w3.org/TR/2013/REC-sparql11-results-json-20130321/, accessed on April 11th, 2022
30. Skjæveland, M.G.: Sgvizler: A javascript wrapper for easy visualization of sparql result sets. In: Extended Semantic Web Conference. pp. 361–365. Springer (2012)
31. Telea, A.C.: Data visualization: principles and practice. CRC Press (2014)
32. Thellmann, K., Galkin, M., Orlandi, F., Auer, S.: Linkdaviz–automatic binding of linked data to visualizations. In: International Semantic Web Conference. pp. 147–162. Springer (2015). https://doi.org/10.1007/978-3-319-25007-6_9
33. Vandenbussche, P.Y., Umbrich, J., Matteis, L., Hogan, A., Buil-Aranda, C.: SPARQLES: Monitoring public SPARQL endpoints. Semantic Web **8**(6), 1049–1065 (Aug 2017). https://doi.org/10.3233/SW-170254
34. Yamamoto, Y., Yamaguchi, A., Splendiani, A.: YummyData: providing high-quality open life science data. Database **2018** (03 2018). https://doi.org/10.1093/database/bay022

35. Zviedris, M., Barzdins, G.: ViziQuer: A Tool to Explore and Query SPARQL Endpoints. In: The Semanic Web: Research and Applications, vol. 6644, pp. 441–445. Springer Berlin Heidelberg, Berlin, Heidelberg (2011). https://doi.org/10.1007/978-3-642-21064-8_31

A Results of the coverage analysis

Result	RDF graph/vocabulary inspection				RDF Summarization		
	RDF graph	Class hierarchy	Property hierarchy	Signature of Properties	Class paths	Property paths	Paths Class → Property → Class
Supported	45.35	38.42	37.71	38.42	41.77	45.11	45.58
HTML	16.71	16.71	16.71	15.75	15.99	15.51	15.27
Service Not Found	14.08	14.08	14.08	14.08	14.32	14.32	14.32
Service Unreachable	6.68	6.68	6.68	6.92	8.59	6.92	6.92
Timeout	6.21	5.97	5.97	5.97	7.88	5.97	5.97
No results	0.72	7.88	8.59	8.59	1.43	1.67	1.19
Invalid Certificate	3.10	3.10	3.10	3.10	3.10	3.10	3.10
Bad Request	1.91	2.15	2.15	2.15	1.91	2.39	2.63
CSV	2.39	2.15	2.15	2.15	2.15	2.15	2.15
Format Not Supported	1.43	1.43	1.43	1.43	1.43	1.43	1.43
Access Unauthorized	0.95	0.95	0.95	0.95	0.95	0.95	0.95
Not W3C compliant	0.48	0.48	0.48	0.48	0.48	0.48	0.48

Table 4. Percentage of SPARQL endpoints per response and per SPARQl query.