

Received July 16, 2019, accepted November 15, 2019, date of publication November 27, 2019, date of current version December 12, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2956345

Linked Data Exploration With RDF Surveyor

GUILLERMO VEGA-GORGOJO^{1,2}, LAURA SLAUGHTER¹, BJØRN MARIUS VON ZERNICHOW³,
NIKOLAY NIKOLOV³, AND DUMITRU ROMAN^{1,3}

¹Department of Informatics, University of Oslo, 0373 Oslo, Norway

²School of Telecommunications Engineering, University of Valladolid, 47011 Valladolid, Spain

³Software and Service Innovation, SINTEF AS, 0373 Oslo, Norway

Corresponding author: Guillermo Vega-Gorgojo (guiveg@tel.uva.es)

This work was supported in part by the Norwegian Research Council through BIGMED (IKT 259055), HealthInsight (NFR 247784/O70), and SIRIUS (NFR 237898), and in part by the European Commission through euBusinessGraph under Grant 732003, in part by the EW-Shopp under Grant 732590, and in part by the Cross-Forest under Grant 2017-EU-IA-0140.

ABSTRACT Linked Data exploration is an essential task in the process of understanding, assessing, and using datasets made available in the Resource Description Framework (RDF) format. Current solutions for exploration of RDF data are mainly targeted at Semantic Web experts, require non-trivial deployments, and do not scale to the increasing amounts of data published in RDF. The lack of simple, intuitive, and efficient solutions for exploring RDF data, especially for lay users, is the main motivation behind the work presented in this paper. We propose RDF Surveyor, an easy-to-use and lightweight tool for exploring RDF datasets. Its visual interface hides the intricacies of Semantic Web technologies from the user, while providing intuitive overviews of datasets, class navigation, and visualization of class instances. Furthermore, RDF Surveyor does not require any installation and can handle large datasets such as DBpedia. We provide a detailed overview of RDF Surveyor and illustrate its capabilities in two different scenarios. We also analyze the uptake, performance and usability of RDF Surveyor, showing its suitability for exploring Linked Data at scale.

INDEX TERMS Linked data, RDF dataset, SPARQL, RDF, exploration tool.

I. INTRODUCTION

With the advent of RDF data [1] across all domains, there is a need for exploring these datasets not only by Semantic Web experts, but also by lay users. However, several studies report that even expert users are severely limited in their ability to obtain a good understanding of the structure and content of RDF datasets, especially if they are large [2], [3]. While Semantic Web experts are used to issuing SPARQL queries [4] in order to explore an unfamiliar repository, this process is rather tedious [5]. Moreover, this is not an option for lay users due to their difficulties with understanding RDF and SPARQL [6].

As early as 2006, [7] identified important user interaction challenges in a pilot deployment of Semantic Web applications; these include severe usability concerns, insufficient guidance to users, and lack of coherence in the visualization of data. Still, a key solution to making sense of RDF data is to provide appropriate visualizations that enable both expert and lay users to understand the structure of a target dataset.

The associate editor coordinating the review of this manuscript and approving it for publication was Mansoor Ahmed.

In this regard, [5] outlines several visualization-driven tasks intended for the exploration of Linked Data: providing suitable entry points, dataset navigation to support exploratory discovery, analysis of data structure, and basic to advanced data querying.

To address the challenge of making sense of RDF data, there is a plethora of tools aiming to provide suitable visualizations (e.g., the tools surveyed in [2], [5], [8], [9]). However, the majority of these proposals are targeted at Semantic Web experts [2], thus severely limiting their adoption by lay users. This is evidenced by the predominance of graph-based interfaces as the default representation for RDF data that has been questioned many times [5], [10], especially for lay users [11]. Furthermore, many of these tools offer limited support for exploration tasks; this is the case of Linked Data browsers such as LodView¹ that present the results of dereferencing URIs in a tabular form and allow link traversal, but do not provide any overview of the dataset nor class navigation. Another important category of visual approaches over RDF data include visual query tools such as PepeSearch [11],

¹<http://lodview.it>

OptiqueVQS [12], and SemFacet [13]; they are specifically purposed for data querying, but all these tools require a non-trivial installation/deployment and are not able to cope with large datasets containing tens of millions of triples or even more.

Our analysis so far shows the need for a tool that supports lay users when exploring an arbitrary RDF dataset. Such a tool should provide an effective visual interface for analyzing the contents, giving an overview of the dataset, and supporting class navigation. Moreover, such a tool should be completely generic and not require any installation to facilitate its usage. Finally, it should work even with large datasets. In this paper we present RDF Surveyor, a novel and lightweight tool for exploring semantic datasets that comply with the requirements stated above.

The rest of this paper is organized as follows: Section II analyses existing approaches for the exploration of Linked Data. Section III presents RDF Surveyor, the tool we have devised for exploring RDF data. Section IV exemplifies two different usage scenarios supported with RDF Surveyor. Section V reports on the use of RDF Surveyor in real practice, including its uptake, a latency analysis, and a usability study with prospective users. Section VI presents a comparison study of RDF exploration tools in an exploration scenario with the English DBpedia. Section VII summarizes the paper and provides ideas for future directions.

II. RELATED WORK

The need of tools to support the exploration of RDF datasets was evident since the earlier years of the Semantic Web [7]. However, preliminary projects addressing this challenge were targeted to either Semantic Web experts or technology enthusiasts willing to put in the time to learn. In recent years, more attention has been given to support the group of “lay users”. [2] defines this lay user as computer literate with some searching skills that enable them to find resources and with interest in everyday tasks such as making comparisons while shopping. The lay user is not a Semantic Web expert and may not have any knowledge of SPARQL, OWL, or RDF. Indeed, this user may quickly give up and move on without appropriate tools to work with Linked Data.

Lay users should be able to easily understand the structure and contents of an RDF dataset by using an appropriate exploration tool. Such a tool should allow them to assess whether a particular dataset is relevant for their needs. Many review articles [2], [3], [9], [14] have outlined specific tasks that might be considered important to users when exploring RDF datasets, e.g., data preview. The Dadzie paper [2] was central in reviewing the requirements for exploring RDF datasets by lay users. Existing solutions typically provide special user interfaces that represent the data visually, while hiding the complexity of RDF, OWL and SPARQL from lay users as much as possible. More recently, [5] summarizes the visualization-driven tasks that should be supported by an RDF exploration tool: to provide suitable entry points; to enable dataset navigation to support exploratory discovery;

to facilitate the analysis of data structure; and to allow basic to advanced data querying. These tasks will be used in the following as a basis for reviewing existing approaches for the exploration of RDF datasets.

A broad type of exploration tools corresponds to Linked Data browsers, i.e., tools that provide interactive support for navigating through or exploring Linked Data [2]. Tabulator [15] and Marbles² are classic examples of Linked Data browsers, while the aforementioned LodView and Phuzzy.link [16] constitute more recent examples. A typical Linked Data browser receives a URI of a resource as input and produces a tabular representation out of the RDF triples in which the input URI participates. As in the case of a regular Web browser, it is easy to jump to another resource by clicking in any of the resource URIs listed. Linked Data browsers are thus conceptually simple and enable the exploration of a dataset by allowing link traversal. However, an exploration session is too dependent on the resource URI provided as input, especially if the dataset contents are very diverse. Further, obtaining a starting URI of a dataset can be frustrating for lay users. Indeed, it is quite difficult to grasp the contents of a dataset with a Linked Data browser since no overview is provided and structural elements – especially classes – are not exploited to arrange the dataset contents.

Visual query editors are another category of tools that can be used for the exploration of RDF datasets. They allow the construction of SPARQL queries through the interaction with visual and manipulable elements. The query language syntax is hidden from the user, while available actions are limited so as to only produce valid queries. The majority of visual query editors are graph-based, exploiting the fact that RDF data can be represented as a graph. The SPARQL syntax is quite apparent in graph-based editors like iSPARQL³ or LuposDate [17], while approaches like NITELIGHT [18], OptiqueVQS [12], or QueryVOWL [19] do a better job hiding the intricacies of SPARQL. However, [10] criticizes the use of graph-based interfaces as the default representation for RDF data – the authors denote “the pathetic fallacy of RDF” to assert that because the data model is a graph, the data should therefore be displayed as a graph. This observation was confirmed in our previous work [20], observing that form-based interfaces are more easily learned and relieve problems with disorientation for mainstream searchers.

Form-based query editors allow the specification of SPARQL queries through the use of text boxes, drop-down menus, radio buttons and other form elements. A prominent subcategory is the one of facet-based interfaces [21, ch. 8] employed to obtain the instances of a class that complies with a set of restrictions. Generally, this type of interfaces is adequate for lay users, so the challenge is to adapt the use of form-based interfaces to RDF datasets. Some proposals have

²<http://mes.github.io/marbles>

³<http://wikis.openlinksw.com/dataspace/owiki/wiki/OATWikiWeb/InteractiveSparqlQueryBuilder>

TABLE 1. Tool functionalities provided for the exploration of RDF datasets.

Tool	Tool type	Entry point	Class navigation	Visualization of individuals	Data querying
Tabulator	Linked Data browser	Textbox (individual URI expected)	No	Yes	No
Marbles	Linked Data browser	Textbox (individual URI expected)	No	Yes	No
LodView	Linked Data browser	Textbox (individual URI expected)	No	Yes	No
Puzzy.link	Linked Data browser	Textbox (SPARQL endpoint + individual URI expected)	No	Yes	No
iSPARQL	Graph-based query editor	SPARQL query	Yes	No	Graphs + SPARQL
LuposDate	Graph-based query editor	SPARQL query	No	No	Graphs + SPARQL
NITELIGHT	Graph-based query editor	Ontology	Yes	No	Graphs + SPARQL
OptiqueVQS	Graph-based query editor	All classes in the dataset	Yes	No	Graphs
QueryVOWL	Graph-based query editor	Textbox (label expected)	No	No	Graphs
Virtuoso Facets	Facet-based query editor	Textbox (label or individual URI expected)	No	Yes	Facets
ExConQuer	Facet-based query editor	All classes in the dataset	Yes	No	Facets
Rhizomer	Facet-based query editor	Upper classes in the dataset	Yes	No	Facets
tFacet	Facet-based query editor	Upper classes in the dataset	Yes	No	Facets
Facet Graphs	Facet-based query editor	Textbox (label expected)	No	No	Facets + graphs
SemFacet	Facet-based query editor	Textbox (label expected)	No	Yes	Facets
PepeSearch	Facet-based query editor	All classes in the dataset	Yes	Yes	Facets
RDF Surveyor	Linked Data exploration tool	Upper classes in the dataset	Yes	Yes	Labels

reproduced faceted search for semantic data by selecting a class of the dataset and then using class properties as facets, as, for example, Virtuoso Facets⁴ and ExConQuer [22]. Note that query expressiveness is somewhat low, since it is not possible to include more than one concept in a query. Due to this, many other attempts have been made to allow for more expressive queries at the cost of ease of use. Pivoting operations allow changing the focus of a query from one class to another, as in Rhizomer [23]. tFacet [24] supports so-called hierarchical facets, providing additional menu forms to include facets from other classes in the user query. Facet Graphs [25] provides a mixed approach that merges graphs with facets, offering a single viewpane that displays both result sets and facets as nodes in a graph visualization. SemFacet [13] just nests all the facets from connected classes in a list. In our previous work we have also proposed PepeSearch [11], a query editor that provides a multi-facet search form automatically constructed from a target RDF dataset.

Table 1 summarizes the capabilities of the tools surveyed in this section. Linked Data browsers are very homogeneous, expecting an individual URI as input and providing visualization of individuals, but no class navigation or data querying capabilities. There are two trends in graph-based query editors, one is to use graphs to complement a SPARQL editor, e.g. iSPARQL, and the other is to completely hide the SPARQL syntax, e.g. QueryVOWL. Class navigation is provided in some cases, while visualization of individuals is seldomly included. While graph-based query editors aim to simplify the creation of complex queries, they are not specifically suited for exploration purposes – [5] criticizes this type of visualization because graphs quickly become very cluttered and illegible, and they tend to underemphasize the contents. In contrast, form-based query editors are more appropriate for exploration purposes, since they are simpler to

use and advanced data querying is not the main requirement. This type of editors typically allow the selection of a class of the dataset and then use class properties as facets in a form interface.

III. OVERVIEW OF RDF SURVEYOR

The literature review in Section II evidenced the need of tools that allow the exploration of RDF datasets without requiring proficiency in SPARQL or RDF. We present here RDF Surveyor, the tool devised for this purpose. RDF Surveyor is a lightweight tool which combines the functionalities of Linked Data browsers and form-based query editors, although data querying is limited to label searches. RDF Surveyor offers a visual interface that hides the intricacies of Semantic Web technologies from the user. The tool supports a set of functionalities for RDF exploration that includes dataset overviews, class navigation, visualization of individuals, and label searches. RDF Surveyor does not require any installation, datasets can be easily configured, and the tool can handle large datasets such as DBpedia.

A. CORE CAPABILITIES

At a first step, the user selects a target dataset by introducing the URI of the SPARQL endpoint and optionally the URI of a named graph. RDF Surveyor will then send a probe to test if the endpoint is up. In case of successful access to the endpoint, an overview of the dataset is presented, consisting of the upper classes and the namespaces found. A namespace identifies the set of names that belong to a single authority, so the list of namespaces can be used to identify the main topics in a repository. Namespaces can then be filtered out – this functionality can be convenient to hide too general, e.g. owl:, or undesired namespaces. With respect to the upper classes, they are aimed to give a good starting point for exploring a dataset. This is a well-known approach in

⁴<http://vos.openlinksw.com/owiki/wiki/VOS/VirtuosoFacetsWebService>

ontology browsers, allowing users to grasp the more general classes available.

When listing classes, RDF Surveyor shows a label in the user language if available or English as a default, the number of direct subclasses, and the number of individuals. Classes are sorted by the number of direct subclasses and individuals to facilitate the identification of the most prominent classes. Importantly, the class hierarchy can be explored by drilling down the subclasses of a listed class. In this way, a user can navigate the classes until a class of interest is found; the focus can then be changed from the dataset overview to a specific class.

After selecting a class of interest, RDF Surveyor prepares a coherent view by including the class URI, label, comment, superclasses, subclasses, and member individuals. Again, subclasses can be navigated and the focus can be changed to another class if desired. With respect to class members, a paginated view is provided for browsing individuals, although it is also possible to search specific individuals by label.

In the case that an individual is selected, RDF Surveyor creates a view with the individual URI, label, comment, corresponding class types, datatype properties, and direct and inverse object properties. The tool also tries to show a picture and a map location by finding a picture URL and geographic coordinates, respectively. With the presented information, the user can jump to other connected individual or class views.

The aforementioned capabilities are intended to support the exploration of a dataset by navigating and browsing the available contents. As an alternative to pure navigation, a search textbox is available for finding classes and individuals by label in the whole dataset.

B. DESIGN PRINCIPLES AND LOGICAL ARCHITECTURE

The design of RDF Surveyor was guided by a set of principles to drive its usefulness and to facilitate its adoption:

- DP1 *Browser-based tool with no server requirements.* Since there is no server component, RDF Surveyor only needs a browser to run, and no further installation is required.
- DP2 *Minimal configuration needed.* The exploration of a dataset only entails a minimal configuration consisting of the URI of the target endpoint and, optionally, the URI of a named graph.
- DP3 *Lazy operation.* Exploration data is obtained on demand by submitting live queries to the target endpoint as required. In this regard, RDF Surveyor does not aim to gather more data than needed to support the user needs.
- DP4 *No assumptions or previous knowledge about a dataset.* RDF Surveyor submits exploratory queries to derive the data structure.
- DP5 *Designed for responsiveness.* In order to meet performance requirements, queries have been carefully crafted for responsiveness, while retrieved data is

TABLE 2. Template URIs used to expose resources in RDF surveyor.

Resource	Template URI
Overview	<code>/?repo={repoURI}&graph={graphURI}</code>
Class	<code>/?repo={repoURI}&graph={graphURI}&class={classURI}</code>
Individual	<code>/?repo={repoURI}&graph={graphURI}&indiv={indivURI}</code>
Search	<code>/?repo={repoURI}&graph={graphURI}&search={keyword}</code>

cached locally in the browser storage during a user session.

- DP6 *Simple form-based user interface.* RDF Surveyor exposes an easy-to-use form-based user interface that completely hides the complexity of SPARQL and RDF from the user.

Complying with all those design principles is quite challenging, especially because we do not make any assumptions about a target dataset (DP4), and the exploration tool has to be responsive (DP5). Lazy operation (DP3) is the key for achieving it: while it is quite tempting to request all the structuring information of a dataset (e.g. the complete class taxonomy), only a subset of it is really needed in order to support a specific user need. We have thus tried to identify the minimum data pieces required to provide the core capabilities described in section III-A. The dataset is queried to obtain the necessary data pieces, using caching and optimized queries to keep latency low.

Figure 1 depicts the logical architecture of RDF Surveyor. Essentially, RDF Surveyor is a web application that exposes resources of repositories. There are four different types of resources: overview, class, individual, and search – they correspond to the capabilities devised for the exploration of a repository (see section III-A). We define the template URIs in Table 2 to expose the different resources in RDF Surveyor.

For a target repository we store the URI of the endpoint and, optionally, the URI of a named graph – this config data is used by the *Query composer* for communicating with the endpoint. *Generators* dynamically create web representations of resources that can then be rendered in the browser. Since RDF Surveyor exposes different types of resources, we include a specialized *Generator* for each resource type, e.g. the *Class view generator* produces web representations of classes.

Generators need some data in order to construct representations of resources. A *Generator* will first look up in the *Data cache* and in case of a missing piece of data it will send a request to the *Query composer*. The *Query composer* will then prepare a suitable SPARQL query and send it to the endpoint, storing the obtained result in the *Data cache* upon receipt, and notifying the calling *Generator* once the data is ready.

The resources exposed by RDF Surveyor are RESTful since every page has a consistent URI that can be safely bookmarked. Note that representations of resources are generated independently, as there are no interdependences among resources. This enables the exploration of large datasets, since RDF Surveyor just obtains the data required to generate a resource representation. In other words, a large dataset

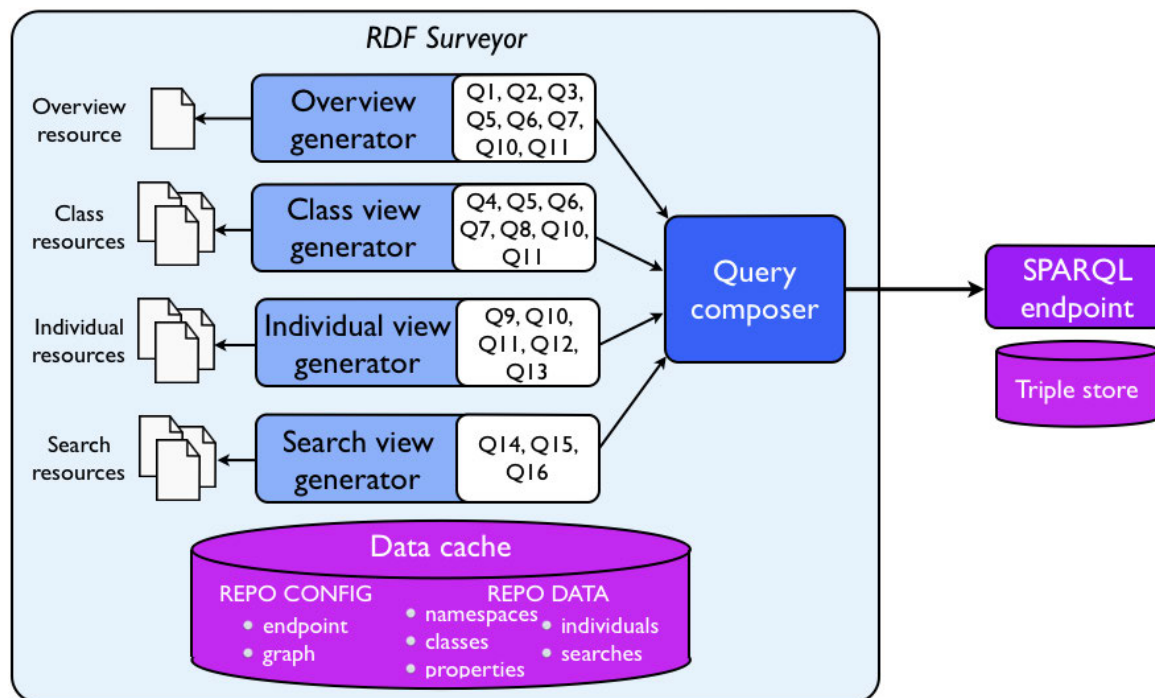


FIGURE 1. Logical architecture of RDF Surveyor.

implies that more resources are available, but the generation of a resource representation is essentially the same for small and large datasets.

C. IMPLEMENTATION OF RDF SURVEYOR

RDF Surveyor is coded in JavaScript to facilitate its deployment as a web application. The source code is available on GitHub.⁵ The user interface is built using the Bootstrap framework,⁶ which enables its usage not only with desktop computers, but also with mobile devices. As described in Section III-B, all exploration data is obtained from the target SPARQL endpoint. For the implementation of the *Data cache* component (see Figure 1) we use the session storage of the browser [26, sec. 11.2.2]. All modern browsers support this feature, although they normally limit the session storage to 10 MB. To address this limitation, RDF Surveyor compresses the data cache before storage. We have been monitoring the effectiveness of this solution since we put it in place, obtaining compression rates around 80–90%. As a result, RDF Surveyor can store 50–100MB of RDF data in the cache – this is quite much for an exploration session and we have no longer experienced the session storage limit.

All the SPARQL queries used in RDF Surveyor are included in Table 3. The *Overview generator* component in Figure 1 relies on the bootstrapping query Q1 to obtain the upper classes in the dataset, i.e. classes with subclasses, but no parent classes. Since the dataset could include isolated

classes, i.e. classes with no subclasses and no parent classes, the bootstrapping query Q2 is employed to gather such isolated classes. For every found occurrence, direct subclasses (query Q5), labels (query Q10) and comments (query Q11) are obtained. In addition, query Q6 is used to assess whether a class has more than one thousand individuals – in case of a negative answer, query Q7 will be used to count its number of individuals.⁷ All this information is employed to prepare a list of upper classes sorted by the number of direct subclasses and members.

The list of upper classes is subsequently analysed to extract a subset of the namespaces employed in the dataset. If the user decides to filter out a namespace, query Q3 is employed to find upper classes that could be unreachable when hiding an undesired namespace, e.g. direct subclasses of `owl:Thing` when filtering out the `owl:` namespace.

The class hierarchy can be navigated by including controls in the user interface to drill down the subclasses of any listed class. In order to present the information about those subclasses, queries Q4, Q5, Q6, Q7, Q10 and Q11 are employed. When a class is selected, the *Class view generator* component (see Figure 1) is in charge of rendering the class information consisting on the class URI, label, comment, breadcrumbs for class navigation, direct superclasses and subclasses, and a paginated view of the class members – the information needed for this latter view is obtained through query Q8.

⁷Counting the members of a class can take a long time if there are many, so query Q6 is used as a shortcut to avoid such expensive computations in the SPARQL endpoint.

⁵<https://github.com/guiveg/rdfsurreyvor>

⁶<http://getbootstrap.com>

TABLE 3. List of SPARQL queries used in RDF surveyor.

ID	Requested data	SPARQL query
Q1	Upper classes	<pre>SELECT DISTINCT ?class WHERE { [] rdfs:subClassOf ?class . filter not exists { ?class rdfs:subClassOf ?super . filter (?super != ?class) } }</pre>
Q2	Isolated classes	<pre>SELECT DISTINCT ?class WHERE { [] rdfs:subClassOf ?class . filter not exists { ?class rdfs:subClassOf ?super . filter (?super != ?class) } filter not exists { ?sub rdfs:subClassOf ?class . filter (?sub != ?class) } }</pre>
Q3	Upper subclasses of classes in namespace {nsuri}	<pre>SELECT DISTINCT ?class WHERE { ?class rdfs:subClassOf ?super . filter (STRSTARTS(STR(?super), "{nsuri}")) filter (!STRSTARTS(STR(?class), "{nsuri}")) }</pre>
Q4	Direct superclasses of {classuri}	<pre>SELECT DISTINCT ?super WHERE { <{classuri}> rdfs:subClassOf ?super . filter not exists { <{classuri}> rdfs:subClassOf ?osuper . ?osuper rdfs:subClassOf ?super . filter (?osuper != <{classuri}>) filter (?osuper != ?super) } filter (?super != <{classuri}>) }</pre>
Q5	Direct subclasses of {classuri}	<pre>SELECT DISTINCT ?sub WHERE { ?sub rdfs:subClassOf <{classuri}> . filter not exists { ?osub rdfs:subClassOf <{classuri}> ?sub rdfs:subClassOf ?osub . filter (?osub != <{classuri}>) filter (?osub != ?sub) } filter (?sub != <{classuri}>) }</pre>
Q6	Classes in {list_of_classuris} with more than one thousand individuals	<pre>SELECT DISTINCT ?class WHERE { ?indiv a ?class . filter (?class in ({list_of_classuris})) } GROUP BY ?class HAVING (count(?indiv) > 1000)</pre>
Q7	Number of individuals for every class in {list_of_classuris}	<pre>SELECT ?class (COUNT(DISTINCT ?indiv) as ?count) WHERE { ?indiv a ?class . filter (?class in ({list_of_classuris})) } GROUP BY ?class</pre>
Q8	Members of {classuri} (paginated with limit {limit} and offset {offset})	<pre>SELECT DISTINCT ?indiv WHERE { ?indiv a <{classuri}> . } LIMIT {limit} OFFSET {offset}</pre>
Q9	Direct types of {indivuri}	<pre>SELECT DISTINCT ?type WHERE { <{indivuri}> a ?type . }</pre>
Q10	Labels in {list_of_uris}	<pre>SELECT DISTINCT ?uri ?label WHERE { ?uri rdfs:label ?label . filter (?uri in ({list_of_uris})) }</pre>

TABLE 3. (Continued.) List of SPARQL queries used in RDF surveyor.

ID	Requested data	SPARQL query
Q11	Comments in {list_of_uris}	SELECT DISTINCT ?uri ?comment WHERE { ?uri rdfs:comment ?comment . filter (?uri in ({list_of_uris})) }
Q12	Properties and values of {individuri} (excluding types, labels and comments)	SELECT DISTINCT ?prop ?val WHERE { <{individuri}> ?prop ?val . filter (?prop != rdf:type) filter (?prop != rdfs:label) filter (?prop != rdfs:comment) }
Q13	Inverse properties and values of {individuri}	SELECT DISTINCT ?prop ?sbj WHERE { ?sbj ?prop <{individuri}> . }
Q14	Non-upper classes containing {keyword} in their label	SELECT DISTINCT ?uri WHERE { ?uri rdfs:subClassOf [] ; rdfs:label ?lab . filter (regex(?lab, "{keyword}", "i")) }
Q15	Individuals containing {keyword} in their label (paginated with limit {limit} and offset {offset})	SELECT DISTINCT ?uri WHERE { ?uri a [] ; rdfs:label ?lab . filter (regex(?lab, "{keyword}", "i")) } LIMIT {limit} OFFSET {offset}
Q16	Members of {classuri} containing {keyword} in their label (paginated with limit {limit} and offset {offset})	SELECT DISTINCT ?uri WHERE { ?uri a <{classuri}> ; rdfs:label ?lab . filter (regex(?lab, "{keyword}", "i")) } LIMIT {limit} OFFSET {offset}

If an individual is selected, queries Q9, Q10, Q11, Q12 and Q13 are used to retrieve the information about such individual. The *Individual view generator* component (see Figure 1) will then render a page with the individual URI, label, comment, a picture (if available in the retrieved data), a geo widget (if a location is found), a list of the class types, literal data, and outgoing and incoming entities.

In addition to browsing, RDF Surveyor supports simple keyword searches of classes and individuals within their labels – query Q14 will be used for finding classes, and query Q15 in the case of individuals. Further, query Q16 is employed in a class page for obtaining the members of a class with a specific keyword.

IV. SAMPLE USAGE SCENARIOS FOR RDF SURVEYOR

We present two different scenarios to illustrate the functionality of RDF Surveyor. The first case corresponds to the standalone use of RDF Surveyor for exploring the DBpedia dataset – the prototypical cross-domain dataset in the Web of Data [27, ch. 3]. The second case corresponds to the integration of RDF Surveyor in a larger platform – DataGraft⁸ [28], [29], a cloud-based platform for data transformation and

publishing. RDF Surveyor has been successfully integrated in DataGraft and is offered as an exploration tool for DataGraft datasets.

A. EXPLORING DBPEDIA CONTENTS

In this scenario we have employed a live version of RDF Surveyor.⁹ In the configuration page we introduce the URIs of the English public SPARQL endpoint¹⁰ and the named graph¹¹ of DBpedia. Figure 2(a) shows an excerpt of the overview page with the namespaces found in the dataset – note that the user has decided to filter out the owl: namespace.

The list of upper classes is presented in Figure 2(b); the user interface includes controls for drilling down the presented classes – this is exemplified with class dbo:Work, showing the list of its subclasses sorted by the number of subclasses and members. The user can go on browsing the class taxonomy, if desired.

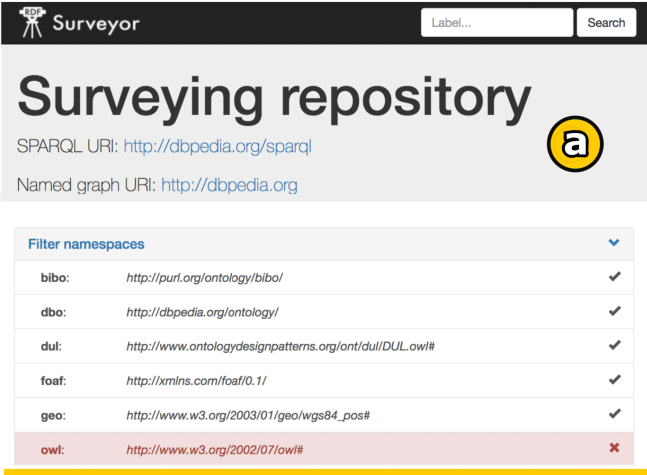
Upon selecting class dbo:Artwork, the page in Figure 2(c) is displayed with the information gathered about this class. The bottom part corresponds to a paginated view

⁸<https://datagraft.io>

⁹<http://tools.sirius-labs.no/rdfsurveyor/>

¹⁰<http://dbpedia.org/sparql>


¹¹<http://dbpedia.org>



Surveying repository
 SPARQL URI: <http://dbpedia.org/sparql>
 Named graph URI: <http://dbpedia.org>

Filter namespaces


bilbo:	http://purl.org/ontology/bibo/	✓
dbo:	http://dbpedia.org/ontology/	✓
dul:	http://www.ontologydesignpatterns.org/ont/dul/DUL_owl#	✓
foaf:	http://xmlns.com/foaf/0.1/	✓
geo:	http://www.w3.org/2003/01/geo/wgs84_pos#	✓
owl:	http://www.w3.org/2002/07/owl#	✗



The Surrender of Breda

http://dbpedia.org/resource/The_Surrender_of_Breda

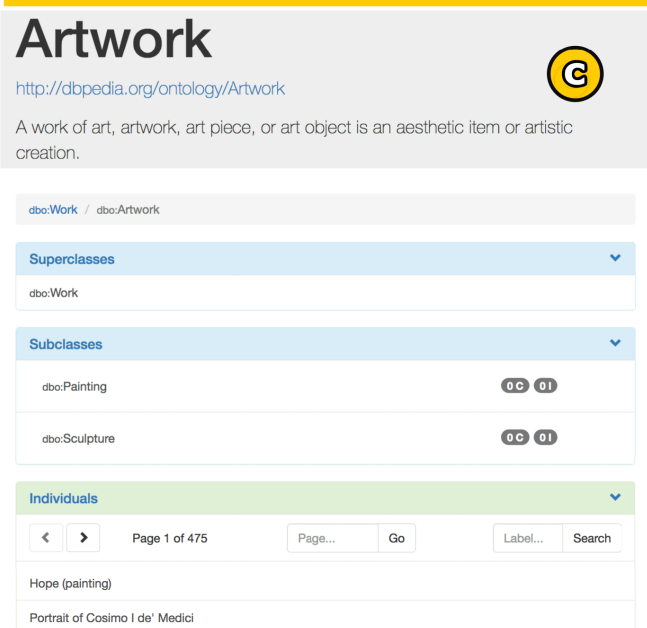
La rendición de Breda (English: The Surrender of Breda, also known as El cuadro de las lanzas or Las lanzas) is a painting by the Spanish Golden Age painter Diego Velázquez. It was completed during the years 1634–35, inspired by Velázquez's visit to Italy with Ambrogio Spinola, the Genoese general who conquered Breda on June 5, 1625. It is considered one of Velázquez's best works. Jan Morris has called it "one of the most Spanish of all pictures".



Types
 dbo:Artwork schema:CreativeWork dbo:Work

Literals

Property	Value
dbo:has abstract	La rendición de Breda (English: The Surrender of Breda, also known as El cuadro de las lanzas or Las lanzas) is a painting by the Spanish Golden Age painter Diego Velázquez. It was completed during the years 1634–35, inspired by Velázquez's visit to Italy with Ambrogio Spinola, the Genoese general who conquered Breda on June 5, 1625. It is considered one of Velázquez's best works. Jan Morris has called it "one of the most Spanish of all pictures".
foaf:name	The Surrender of Breda
dbp:title	The Surrender of Breda
dbo:Wikipedia page ID	14237621
dbo:Wikipedia revision ID	666714018



Artwork

<http://dbpedia.org/ontology/Artwork>

A work of art, artwork, art piece, or art object is an aesthetic item or artistic creation.

dbo:Work / dbo:Artwork

Superclasses
 dbo:Work

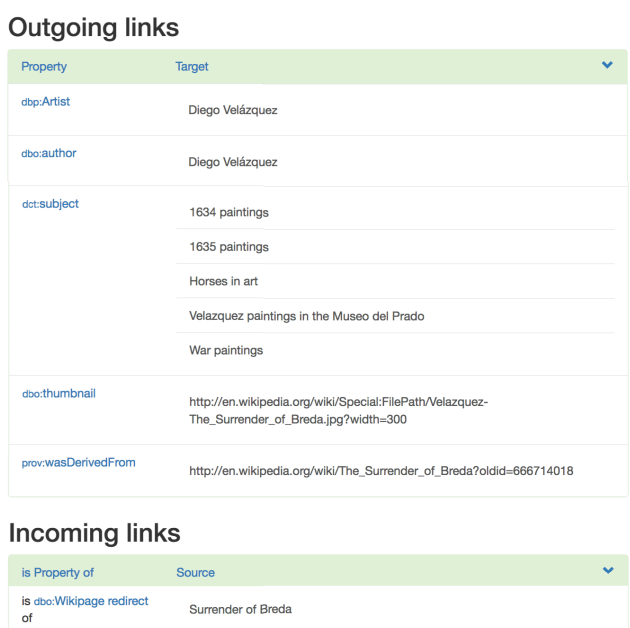
Subclasses

dbo:Painting	0 C 0 I
dbo:Sculpture	0 C 0 I

Individuals

Page 1 of 475

Hope (painting)
Portrait of Cosimo I de' Medici



Outgoing links

Property	Target
dbp:Artist	Diego Velázquez
dbo:author	Diego Velázquez
dc:subject	1634 paintings 1635 paintings Horses in art Velazquez paintings in the Museo del Prado War paintings
dbo:thumbnail	http://en.wikipedia.org/wiki/Special:FilePath/Velazquez-The_Surrender_of_Breda.jpg?width=300
prov:wasDerivedFrom	http://en.wikipedia.org/wiki/The_Surrender_of_Breda?oldid=666714018

Incoming links

is Property of	Source
is dbo:Wikipedia redirect of	Surrender of Breda

FIGURE 2. Sample snapshots of RDF Surveyor when exploring DBpedia: (a) Excerpt of the namespaces for filtering. (b) Overview of the upper classes. (c) Description of the Artwork class. (d) Description of The Surrender of Breda individual.

of the members of `dbo:Artwork`, including controls for navigating pages and a search textbox for finding members by label.

When an individual is selected, a new page will be prepared, e.g., Figure 2(d) for “The Surrender of Breda” painting. Collected information about the referred individual is presented, as explained in Section III. The user can jump to other individual or class pages by clicking the corresponding links.

It is worth mentioning that all pages include a search textbox for finding classes and individuals by label in the whole dataset (see the navbar in Figure 2(a)).

B. EXPLORING DATAGRAFT DATASETS

RDF Surveyor is available with all the SPARQL endpoints that have been created or registered in the DataGraft platform. This tool is embedded in the page that provides descriptions and metadata about a SPARQL endpoint, as well as an access point to issue SPARQL queries. We implemented some minor customizations in RDF Surveyor in order to make a seamless integration with DataGraft. Specifically, upon selecting a SPARQL endpoint in DataGraft, the overview page generated by RDF Surveyor is automatically injected in the view. Thereafter, any queries generated by RDF Surveyor are issued to that endpoint. In this way, no configuration is needed from the user and the dataset contents can be directly explored.

We illustrate this integration with a dataset containing company information. The data can be published using the wizards that are provided in the DataGraft portal dashboard as shown in Figure 3(a) and (b). After the data has been published, and the SPARQL endpoint has successfully been loaded, a user may choose to make it publicly available to other users of the platform. An interested user can go to the SPARQL endpoint and explore the contents. She can access the functionality of RDF Surveyor by clicking on the “Browsing” option in the SPARQL endpoint page, as shown in Figure 3(c). This page corresponds to the dataset overview, presenting the available namespaces and the upper classes – similarly to Figure 2(a). This dataset contains five upper classes, namely addresses, identifiers, organizations, sites and registers. By clicking on the “RegisteredOrganisation” class, the user can browse the list of companies published in the dataset (Figure 3(d)) and select an organization of interest (Figure 3(e)). She can continue exploring the contents or perform a global search or search for individuals within a class, as in the standalone version of RDF Surveyor.

Therefore, the integration of RDF Surveyor in DataGraft allows data publishers to showcase their datasets to potential data consumers. Correspondingly, data users are able to gain insight into the structure and content of a repository, or to simply find items of interest without any prerequisite for knowledge of SPARQL.

V. RDF SURVEYOR IN PRACTICE

In this section we first assess the uptake of RDF Surveyor by analyzing the tracking data of the standalone and the

TABLE 4. Uptake of RDF surveyor.

Item	Standalone	DataGraft
# of users	151	375
# of sessions	287	531
# of repository visits	472	694
# of unique repositories	47	61
# of class visits	801	256
# of unique classes	242	76
# of individual visits	604	309
# of unique individuals	320	155

DataGraft versions (see Section IV). Then, we present the latencies measured when performing the different operations supported by RDF Surveyor. This section ends with a usability study carried out with 14 data scientists who volunteered to test the DataGraft version of RDF Surveyor.

A. UPTAKE OF RDF SURVEYOR

In order to track the usage of RDF Surveyor we employ the Google Analytics platform¹² both in the standalone and the DataGraft versions of the tool. Table 4 summarizes the collected data (obtained in November 2019).

Concerning the standalone version of RDF Surveyor, 151 users have used this tool in 287 sessions to explore 47 unique repositories. 54% of the visits surveyed the DBpedia dataset, 8% the LinkedGeoData knowledge base,¹³ 7% the Nobel prize repository,¹⁴ and 3% the WarSampo Knowledge Graph¹⁵ – the remaining repositories received less than 2% of the total traffic. Since most of the sessions surveyed DBpedia, the most visited classes and individuals correspond to that repository.

The DataGraft version of RDF Surveyor has received more traffic: 375 users in 531 sessions to explore 61 unique repositories. Traffic was less biased than in the standalone case since 17 datasets received at least 2% of the visits. The repositories in DataGraft have a smaller size than DBpedia or LinkedGeoData, so users comparatively explored a smaller number of classes and individuals.

B. LATENCY OF RDF SURVEYOR

We have also employed Google Analytics to register the latency of the different operations carried out with RDF Surveyor. This performance data is important to assess whether our approach is viable for exploring RDF datasets given that data is queried on demand without any bootstrapping procedure (see Section III). Table 5 presents the collected data as obtained in November 2019.

Beginning with the analysis of the standalone version of RDF Surveyor, the repository page load operation takes 2.42 seconds in average. This operation is quite common (17%) and relies on the performance of queries Q1 and Q2

¹²<https://www.google.com/analytics>

¹³<http://linkedgeo.org>

¹⁴<http://data.nobelprize.org>

¹⁵<http://www.ldf.fi/dataset/warsa>

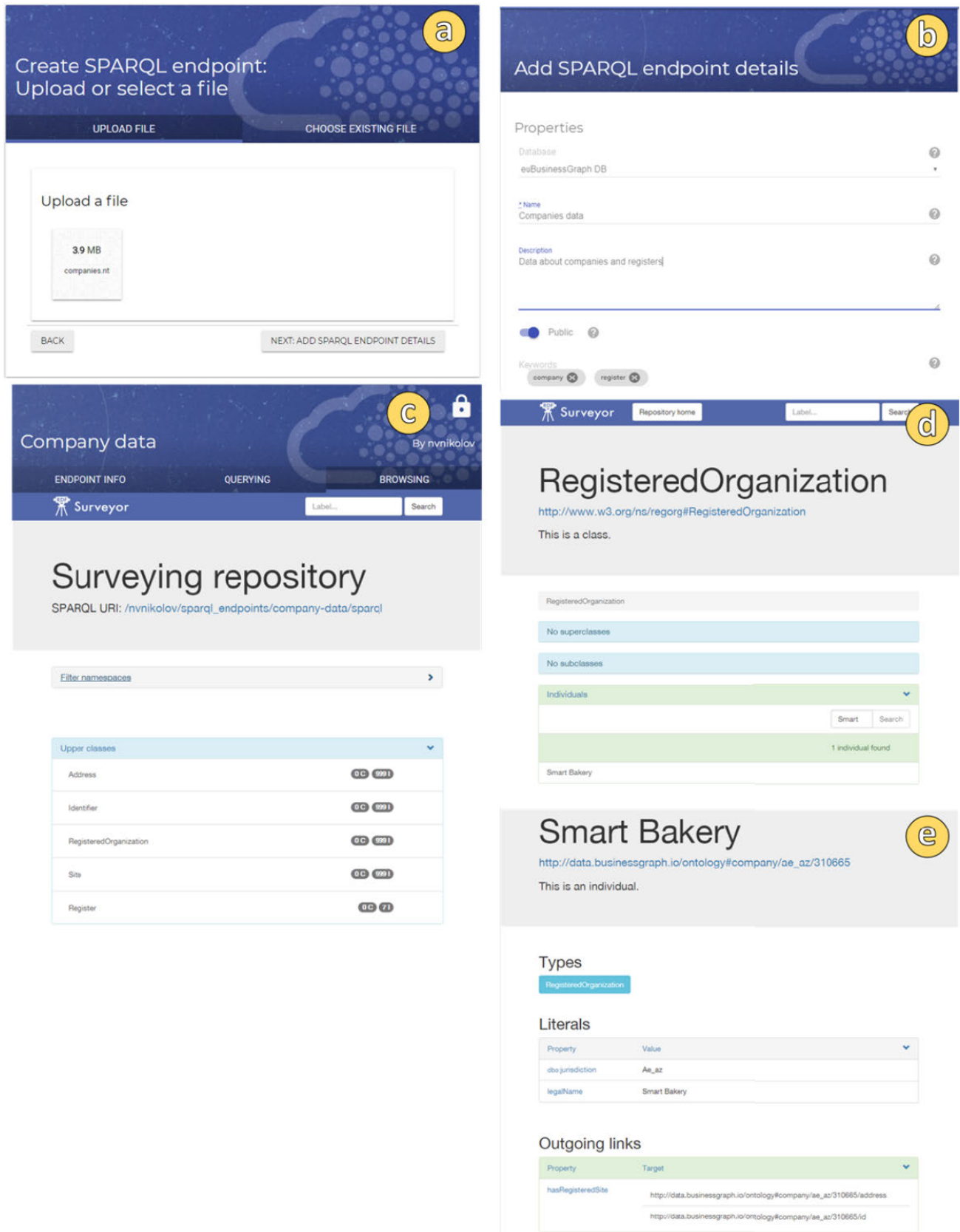


FIGURE 3. Using RDF Surveyor to explore SPARQL endpoints on DataGraft: (a) Uploading company data to DataGraft using the SPARQL endpoint publication wizard. (b) Specifying SPARQL endpoint details. (c) Surveying the published SPARQL endpoint in the Browsing tab. (d) Searching for specific organisations in the SPARQL endpoint (e) Displaying details about the organization.

TABLE 5. Latency of RDF surveyor.

Operation	Time (avg.)	
	Standalone	DataGraft
Repository page load	2.42 s.	1.65 s.
Class page load	0.42 s.	0.36 s.
Individual page load	0.32 s.	0.47 s.
Disable namespace	5.36 s.	0.51 s.
Enable namespace	0.03 s.	0.01 s.
Class expansion	0.40 s.	0.62 s.
Search individuals within a class	1.60 s.	0.36 s.
Global search of classes and individuals	3.71 s.	0.41 s.

(see Table 3) which are dependent on the size of the dataset schema – quite big in the case of DBpedia. The remaining bulk of the workload corresponds to class page load (30%), individual page load (23%), and class expansion (9%) operations – all these cases take less than one second which is the limit for the user's flow of thought to stay uninterrupted [30]. The disable namespace operation is the slowest due to the expensive query Q3 with the `STRSTARTS` function – fortunately, only used in a 2% of the cases. Keyword searches are also slow (queries Q15 and Q16), especially global searches (4% of the workload).

With respect to the DataGraft version of RDF Surveyor, the latency figures are significantly better because the target repositories have a smaller size (up to 1.4 million triples) than the ones employed in the standalone version (note that the English DBpedia contains 1.7 billion triples [31]). Every operation but the repository page load takes less than one second on average, so the flow of exploration can be conducted without noticeable interruptions.

C. USABILITY STUDY

We have carried out a usability study of RDF Surveyor with 14 data scientists who volunteered to participate and gave their consent to use their study data for research purposes. Four of them have almost no experience with Linked Data and no knowledge of SPARQL; eight participants have been involved 1–3 years with Linked Data and some basic knowledge of SPARQL; the remaining two have been working with Linked Data for more than 3 years and have good knowledge of SPARQL.

Participants used the version of RDF Surveyor integrated in DataGraft with a dataset about cadastral properties in Norway managed by Statsbygg,¹⁶ which is key advisor of the Norwegian Government in construction and property affairs, building commissioner, property manager and property developer. Statsbygg publishes cross-sectorial data about public properties to assess and report the efficiency and sustainability of the government's civil estate in a report called State of Estate (SoE) [32]. The ontology for the SoE report includes RDF domain vocabularies that have been developed within the proDataMarket¹⁷ project under the umbrella of

the proDataMarket ontology [33] (e.g., proDataMarket SoE Vocabulary,¹⁸ proDataMarket Cadaster Vocabulary,¹⁹) while the SoE dataset [34] contains 1.3 million triples. Participants did not have previous experience with cadastral information, and were given a short introduction (~20 minutes) about RDF Surveyor. Afterwards they were required to perform several exploration tasks with this dataset, namely:

- 1) Scenario 1: Show the state-owned cadastral parcels in a given municipality
- 2) Scenario 2: Find the owner/lessor of a state-owned property with a national cadastral ID 0214/100/11/0/0
- 3) Scenario 3: Find the cadastral parcels and buildings owned by a central government organization such as METEOROLOGISK INSTITUTT with organization number 971274042

Participants were requested to fill a questionnaire after testing RDF Surveyor. The employed questionnaire has three sections: the first one corresponds to the System Usability Score (SUS) [35], a popular questionnaire for evaluating the usability of system interfaces; the second section consisted of two open-ended questions about the strengths and weaknesses of RDF Surveyor, namely *What did you like about RDF Surveyor?* and *What did you not like about RDF Surveyor?*; the third section included rating scale questions about their overall impression of the tool.

We computed the SUS scores for the participants' responses, obtaining 65 in average with a standard deviation of 16. This is a fair score, given that SUS scores range from 0 to 100. According to the grading scale interpretation of SUS scores in [36, ch. 8], RDF Surveyor was graded with a C. Interestingly, the group of participants less involved in Linked Data gave the lowest scores (51 in average), while the group with more experience in Linked Data and SPARQL ranked RDF Surveyor with a SUS score of 80 in average.

The analysis of the second part of the questionnaire helped us to further understand the SUS results. As this part consisted of open-ended questions, we identified several themes and categorised the answers. The main findings are included in Table 6, along with their level of support and a sample participants' comment for each finding. On the positive side, participants in the study stressed the capability of the tool for exploring RDF datasets – this is indeed the main purpose of RDF Surveyor. They also considered the tool easy to use and emphasised its user experience. Further, three participants rated RDF Surveyor as lightweight.

Regarding limitations, some participants criticised the need to know the structure of the dataset. We believe this problem was due to the employed dataset: participants were not familiar with the domain of cadastral information and the dataset did not include labels or comments that are crucial to grasp the data structure – this may have negatively affected the obtained SUS score. In addition, participants reported some usability problems and lack of intuitiveness. These

¹⁶<https://www.statsbygg.no>

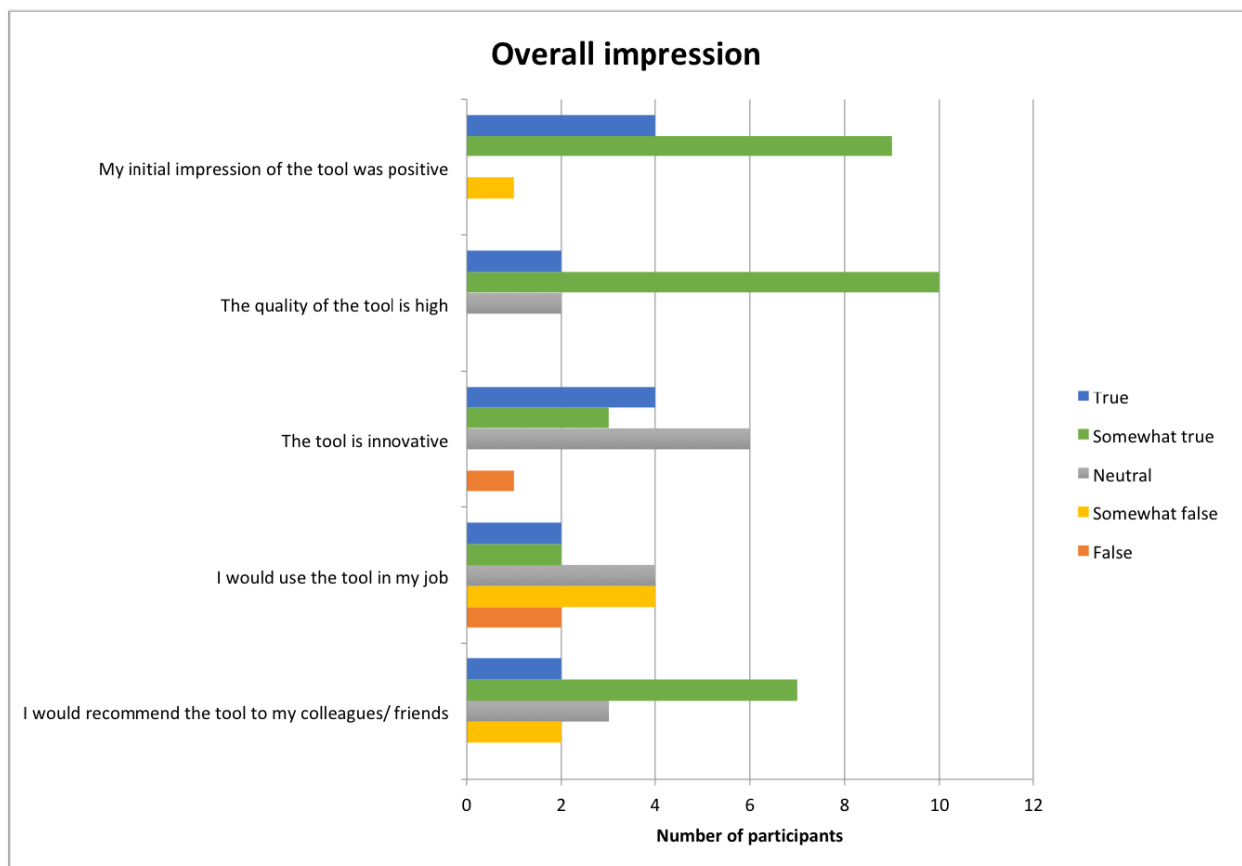
¹⁷<https://prodatamarket.eu>

¹⁸<http://vocabs.datagraft.net/proDataMarket/0.1/SoE>

¹⁹<http://vocabs.datagraft.net/proDataMarket/0.1/Cadaster>

TABLE 6. Strengths and weaknesses of RDF surveyor obtained from the second part of the questionnaire.

Point	Support	Sample comment
+ Good exploration tool	5/14	<i>Easy to explore the rdf repository without much requirement on the RDF knowledge [P11]</i>
+ Easy to use	5/14	<i>It is easy to start using [P9]</i>
+ User experience	4/14	<i>General look and feel was familiar and confidence-inspiring [P8]</i>
+ Lightweight, runs client side	3/14	<i>Easy to use, no technical knowledge required, works real-time, lightweight, easy to understand [P13]</i>
- Requires knowledge of the dataset structure	3/14	<i>Not easy to do the tasks if one doesnt know the structure of the RDF [P14]</i>
- Usability problems	3/14	<i>It was not clear to me that I was expected to click next to the arrow but not on the arrow to get to individuals [P4]</i>
- Not intuitive	3/14	<i>Easy to misunderstand [P12]</i>
- No summary of results	2/14	<i>When you get a list of results you have to guess where to start or have to go thru all items in the list which is cumbersome if list of results is larger than few items [P9]</i>
- Somewhat slow	1/14	<i>It was a bit slow in some cases [P5]</i>

**FIGURE 4.** Overall impression of RDF surveyor by the participants in the usability study.

problems may be addressed by including tooltips and help functionalities, as well as with user training. Besides, two participants requested better support for summarising results and one considered RDF Surveyor a bit slow.

To conclude the analysis of the conducted usability study, Figure 4 graphically depicts the overall impression of RDF Surveyor, obtained from the answers to the third part of the questionnaire. These results are generally very positive, especially regarding the initial impression and the tool quality. Nevertheless, the question about adoption (*I would use the tool in my job*) was not so supportive, especially in the group of participants less involved in Linked Data.

VI. COMPARISON STUDY IN AN EXPLORATION SCENARIO

After reporting the use of RDF Surveyor in real practice, we present here a comparison study of the tools surveyed in Section II. For this purpose, we use the exploration scenario described in Section IV-A as it depicts a succinct and understandable exploration path that exploits the English DBpedia. This dataset is well-known, but also very challenging due to the comprehensiveness of the DBpedia ontology and the volume of the dataset. As a result, this scenario can serve to detect performance and scalability issues of Linked Data exploration tools.

TABLE 7. Tool support in the “Exploring DBpedia contents” scenario.

Tool	Installation requirements	Dataset selection	Working	Working with DBpedia	Scenario workarounds	Performance and scalability issues found
Tabulator	None (browser)	Individual URI	Yes	No (not data found)	–	–
Marbles	Server web app installation	Individual URI	No (installation failed)	–	–	–
LodView	None (browser)	Individual URI + optional SPARQL endpoint	Yes	Yes	Class navigation and data querying are not supported No members of <code>dbo:Artwork</code> were listed Only available option was to use <code>dbr:The_Surrender_of_Breda</code> as input	Not observed
Phuzzy.link	None (browser)	Individual URI + SPARQL endpoint	Yes	No (CORS request rejected)	–	–
iSPARQL	None (browser)	SPARQL endpoint	Yes	Yes	Difficulties using the graph editor to visually create a query, we just employed the plain SPARQL editor	Long delays and errors trying to navigate the class taxonomy Graph editor unhelpful for dataset exploration Cannot be used with an external dataset
LuposDate	Java Web Start application	Packed datasets	Yes	No	–	–
NITELIGHT	Unknown	Load ontology + SPARQL endpoint	No (not available)	–	–	–
OptiqueVQS	Optique platform installation	Load ontology + SPARQL endpoint	Yes	No (published version does not allow SPARQL endpoint selection)	–	Need to load the ontology used in the dataset in advance Providing a flat list of classes as an entry point is problematic with large ontologies The tested version of the tool does not allow the selection of an alternative dataset
QueryVOWL	None (browser)	Bound to DBpedia	Yes	Yes	No class navigation so we had to guess the right class: <code>dbo:Artwork</code> No visualization of individuals	Limited to Virtuoso triplestores Need to install a package in the triplestore
Virtuoso Facets	Virtuoso package installation	Bound to a Virtuoso triplestore	Yes	Yes	No class navigation so we had to guess the right class: <code>dbo:Artwork</code>	–
ExConQuer	Unknown	SPARQL endpoint	No (not available)	–	–	–
Rhizomer	Server web app installation	Load dataset or datastore config	Yes	No (requires database credentials of the DBpedia Virtuoso)	–	Need to load the dataset or know the database credentials
tFacet	None (browser)	SPARQL endpoint	Yes	No (not data found)	–	–
Facet Graphs	None (browser)	SPARQL endpoint	Yes	No (not data found)	–	–
SemFacet	Server web app installation	Load dataset	No (installation failed)	–	–	Need to load the dataset Cannot be used with an external dataset
PepeSearch	Schema analyzer installation	SPARQL endpoint selection	Yes	No (analyzer timed out)	–	Need to run the schema analyzer in advance Providing a flat list of classes as an entry point is problematic with large ontologies
RDF Surveyor	None (browser)	SPARQL endpoint selection	Yes	Yes	–	Not observed

In this comparison, we first analyze the feasibility of every tool for exploring the DBpedia contents by identifying the installation requirements and dataset selection mechanism, as well as testing whether a tool works with DBpedia. Afterwards, we try to reproduce the exploration path of the scenario, taking note of the workarounds needed to complete it, and detecting performance and scalability issues in this process.

Table 7 presents the results of this comparison study. The main findings are the following:

- We were able to run 13 of the 17 tools analyzed. With respect to the non-working tools, two of them were not available, while in the remaining two cases we were not able to successfully deploy the provided web applications in a Tomcat container (we even tried using the specific Tomcat versions mentioned in the tool documentation).
- 8 of the 13 working tools were not capable of working with DBpedia: in three cases no data was retrieved from the DBpedia SPARQL endpoint, two other tools did not allow the selection of an external dataset (like DBpedia), another required database credentials, and the remaining two failed when accessing DBpedia.
- LodView was the only Linked Data browser working with DBpedia. However, this tool is quite limited for exploring Linked Data since (1) it requires an individual URI as input; (2) no class browsing is supported; and

(3) using a class URI, e.g. `dbo:Artwork`, as input does not list its members. Therefore, the only available option was to provide the URI of the desired individual (`dbr:The_Surrender_of_Breda`). It was then possible to navigate to other related individuals from there and we did not observe performance or scalability problems in this case.

- iSPARQL allows the construction of SPARQL queries through a graph editor. An ontology pane is provided for navigating the classes that can be used as building blocks of a visual query. Unfortunately, the initial query to get the ontology classes of DBpedia timed out and an error dialog was shown. The graph editor was not very helpful since we could not select specific classes from the DBpedia ontology to constrain nodes. Instead, we just used the plain SPARQL editor to prepare a query asking for individuals of class `dbo:Artwork` with a label filter with the text “Breda”.
- QueryVOWL is a graph-based query editor. This tool does not support class browsing, so we used a textbox to find the class `dbo:Artwork`. We assigned this class to a node, obtaining a list of individuals of the corresponding class. We then introduced the term “Breda” in a filter box in order to obtain the desired individual (`dbr:The_Surrender_of_Breda`). QueryVOWL does not offer a dedicated view for individuals, a side

bar just shows a label and a comment. We did not experience performance problems using the tool. Concerning scalability, the tested version is bound to DBpedia, so it cannot be used with other datasets.

- Virtuoso Facets does not support class navigation, so we searched the class `dbo:Artwork` using the “Entity Label Lookup” textbox. We then selected the option “Start New Facet” and then filtered the entities with the text “Breda” in order to find the desired individual (`dbr:The_Surrender_of_Breda`). Virtuoso Facets also gives a tabular view of a selected individual, although no picture or geo widget is provided. We did not observe performance issues in this process. Scalability is limited, given that it only works with Virtuoso datasets, requiring the installation of the Virtuoso Facets package in the target repository. Therefore, this tool cannot be used with many datasets – although it is available in the case of the English DBpedia.

Overall, there are only 4 working tools besides RDF Surveyor that worked with DBpedia. However, support for exploration is somewhat limited since none of the alternatives to RDF Surveyor allow class navigation (three by design, while iSPARQL failed). Another important feature for dataset exploration is visualization of individuals, but only LodView and Virtuoso Facets include this functionality. Of the tools surveyed, we only observed performance problems in the case of iSPARQL when trying to navigate the taxonomy of DBpedia classes. Regarding scalability concerns, Virtuoso Facets can only be used with Virtuoso triplestores (if activated by the provider), while QueryVOWL is bound to the DBpedia dataset.

VII. SUMMARY AND OUTLOOK

Despite the increasing volumes of Linked Data on the Web, the use of such data is hindered by a number of important challenges. Exploration of Linked Data should allow prospective users to understand, assess, and select available datasets. This is especially crucial for the take-up of Linked Data by lay users that are not experts in the technicalities of Semantic Web technologies, and are in need for simple, intuitive, and efficient solutions for exploring RDF datasets.

We identified a gap in the literature (and practice) around visualization and explorations of Linked Data – the fact that there is a lack of a simple, yet intuitive approach and corresponding system for exploring RDF data, while at the same time being able to cope with the increase scale of the datasets published as Linked Data. To alleviate this gap we proposed RDF Surveyor – a novel and lightweight tool for exploring RDF datasets targeted to lay users. In this paper we presented an overview of RDF Surveyor focusing on its design principles, core functionalities, system architecture, and implementation details. The tool is available as open source and has been successfully integrated within the cloud-based platform DataGraft. Indeed, there is a significant uptake of RDF Surveyor and the tool is able to cope with large-scale datasets such as DBpedia while keeping latency

low. We also report the results of a usability study that helped to validate the design assumptions of RDF Surveyor and indicates its suitability as a practical solution for exploring Linked Data at scale.

As part of our future work we plan to improve the tool according to the feedback received in the usability study (see Section V-C) and explore mechanisms for a wider dissemination of RDF Surveyor to communities in various domains that could potentially benefit from the the available Linked Data, and implicitly from the use of RDF Surveyor for Linked Data exploration. We also aim to improve the data querying capabilities of the tool by supporting facet search.

REFERENCES

- [1] R. Cyganiak, D. Wood, and M. Lanthaler. (Feb. 2014). RDF 1.1 concepts and abstract syntax. W3C Recommendation. Accessed: Jan. 2019. [Online]. Available: <http://www.w3.org/TR/rdf11-concepts/>
- [2] A.-S. Dadzie and M. Rowe, “Approaches to visualising linked data: A survey,” *Semantic Web*, vol. 2, no. 2, pp. 89–124, 2011.
- [3] J. Klímek, M. Nečaský, B. Kostov, M. Blaško, and P. Křemen, “Efficient exploration of linked data cloud,” in *Proc. 4th Int. Conf. Data Manage. Technol. Appl.*, Colmar, France, 2015, pp. 255–261.
- [4] W3C SPARQL Working Group. (Mar. 2013). *SPARQL 1.1 Overview*. Accessed: Jan. 2019. [Online]. Available: <http://www.w3.org/TR/sparql11-overview/>
- [5] A.-S. Dadzie and E. Pietriga, “Visualisation of linked data—Reprise,” *Semantic Web*, vol. 8, no. 1, pp. 1–21, 2017.
- [6] A. Khalili, A. Loizou, and F. van Harmelen, “Adaptive linked data-driven Web components: Building flexible and reusable semantic Web interfaces,” in *Proc. 15th Int. Semantic Web Conf. (ISWC)*, Kobe, Japan, 2016, pp. 677–692.
- [7] T. Heath, J. Domingue, and P. Shabajee, “User interaction and uptake challenges to successfully deploying Semantic Web technologies,” in *Proc. 3rd Int. Semantic Web User Interact. Workshop 5th Int. Semantic Web Conf. (SWUI)*, Athens, GA, USA, 2006.
- [8] N. Marie and F. Gandon, “Survey of linked data based exploration systems,” in *Proc. 3rd Int. Workshop Intell. Explor. Semantic Data (IESD)*, Riva Del Garda, Italy, Oct. 2014, pp. 1–12.
- [9] J. Klímek, P. Škoda, and M. Nečaský, “Survey of tools for linked data consumption,” *Semantic Web*, vol. 10, no. 4, pp. 665–720, 2018.
- [10] M. Schraefel and D. Karger, “The pathetic fallacy of RDF,” in *Proc. 3rd Int. Semantic Web User Interaction Workshop (SWUI) 5th Int. Semantic Web Conf.*, Athens, GA, USA, 2006.
- [11] G. Vega-Gorgojo, M. Giese, S. Heggstøyl, A. Soylu, and A. Waaler, “PepeSearch: Semantic data for the masses,” *PLoS ONE*, vol. 11, no. 3, 2016, Art. no. e0151573.
- [12] A. Soylu, M. Giese, E. Jimenez-Ruiz, G. Vega-Gorgojo, and I. Horrocks, “Experiencing OptiqueVQS: A multi-paradigm and ontology-based visual query system for end users,” *Universal Access Inf. Soc.*, vol. 15, no. 1, pp. 129–152, 2016.
- [13] M. Arenas, B. C. Grau, E. Kharlamov, V. S. Marciuška, and D. Zheleznyakov, “Faceted search over RDF-based knowledge graphs,” *J. Web Semantics*, vol. 37, pp. 55–74, Mar. 2016.
- [14] N. Bikakis, *Big Data Visualization Tools*. Cham, Switzerland: Springer, 2018.
- [15] T. Berners-Lee, Y. Chen, L. Chilton, D. Connolly, R. Dhanaraj, J. Hollenbach, A. Lerer, and D. Sheets, “Tabulator: Exploring and analyzing linked data on the semantic Web,” in *Proc. 3rd Int. Semantic Web User Interact. Workshop (SWI)*, Nov. 2006, pp. 1–16.
- [16] B. Regalia, K. Janowicz, and G. Mai, “Phuzzy.link: A SPARQL-powered client-sided extensible semantic Web browser,” in *Proc. 3rd Int. Workshop Vis. Interact. Ontologies Linked Data (VOILA) 16th Int. Semantic Web Conf.*, Vienna, Austria, Oct. 2017, pp. 1–11.
- [17] J. Groppe, S. Groppe, A. Schleifer, and V. Linnemann, “LupusDate: A semantic Web database system,” in *Proc. 18th ACM Conf. Inf. Knowl. Manage. (CIKM)*, Hong Kong, Nov. 2009, pp. 2083–2084.
- [18] A. Russell, P. R. Smart, D. Braines, and N. R. Shadbolt, “NITELIGHT: A graphical tool for semantic query construction,” in *Proc. Semantic Web User Interact. Workshop (SWUI)*, Florence, Italy, Apr. 2008, pp. 1–10.

- [19] F. Haag, S. Lohmann, S. Siek, and T. Ertl, "QueryVOWL: Visual composition of SPARQL queries," in *Proc. 12th Eur. Semantic Web Conf. (ESWC)*, Portoroz, Slovenia, Jun. 2015, pp. 1–5.
- [20] G. Vega-Gorgojo, L. Slaughter, M. Giese, S. Heggestøyl, A. Soylu, and A. Waaler, "Visual query interfaces for semantic datasets: An evaluation study," *J. Web Semantics*, vol. 39, pp. 81–96, Aug. 2016.
- [21] M. Hearst, *Search User Interfaces*. Cambridge, U.K.: Cambridge Univ. Press, 2009.
- [22] J. Attard, F. Orlandi, and S. Auer, "ExConQuer: Lowering barriers to RDF and Linked Data re-use," *Semantic Web*, vol. 9, no. 2, pp. 241–255, 2018.
- [23] J. M. Brunetti, R. García, and S. Auer, "From overview to facets and pivoting for interactive exploration of semantic Web data," *Int. J. Semantic Web Inf. Syst.*, vol. 9, no. 1, pp. 1–20, 2013.
- [24] S. Brunk and P. Heim, "tFacet: Hierarchical faceted exploration of semantic data using well-known interaction concepts," in *Proc. Int. Workshop Data-Centric Interact. Web (DCI)*, Lisbon, Portugal, Sep. 2011, pp. 31–36.
- [25] P. Heim, T. Ertl, and J. Ziegler, "Facet graphs: Complex semantic querying made easy," in *The Semantic Web: Research and Applications (Lecture Notes in Computer Science)*, vol. 6088, L. Aroyo, G. Antoniou, E. Hyvönen, A. t. Teije, H. Stuckenschmidt, L. Cabral, and T. Tudorache, Eds. Springer, Jan. 2010, pp. 288–302.
- [26] Web Hypertext Application Technology Working Group. (Jun. 2019). *HTML Living Standard WHATWG (Apple, Google, Mozilla, Microsoft)*. Accessed: Jul. 2019. [Online]. Available: <https://html.spec.whatwg.org/>
- [27] T. Heath and C. Bizer, *Linked Data: Evolving the Web Into a Global Data Space*. San Rafael, CA, USA: Morgan & Claypool, 2011.
- [28] D. Roman, N. Nikolov, A. Putlier, D. Sukhobok, B. Elvesæter, A. Berre, X. Ye, M. Dimitrov, A. Simov, M. Zarev, R. Moynihan, B. Roberts, I. Berlocher, S. Kim, T. Lee, A. Smith, and T. Heath, "DataGraft: One-stop-shop for open data management," *Semantic Web*, vol. 9, no. 4, pp. 393–411, 2018.
- [29] D. Roman, M. Dimitrov, N. Nikolov, A. Putlier, D. Sukhobok, B. Elvesæter, A. Berre, X. Ye, A. Simov, and Y. Petkov, "DataGraft: Simplifying open data publishing," in *Proc. 13th Eur. Semantic Web Conf. (ESWC)*, in *Lecture Notes in Computer Science*, vol. 9678, H. Sack, G. Rizzo, N. Steinmetz, D. Mladenčić, S. Auer, and C. Lange, Eds. Heraklion, Greece: Springer, May 2016, pp. 101–106.
- [30] J. Nielsen, "Response times: The 3 important limits," Nielsen Norman Group, Fremont, CA, USA, 1993. Accessed: Jul. 2018. [Online]. Available: <https://www.nngroup.com/articles/response-times-3-important-limits/>
- [31] S. Praetor. (Jul. 2017). *New DBpedia Release—2016-10*. Accessed: Jul. 2018. [Online]. Available: <http://blog.dbpedia.org/2017/07/04/new-dbpedia-release-2016-10/>
- [32] L. Shi, B. E. Pettersen, I. Østhassel, N. Nikolov, A. Khorramhonam, A. J. Berre, and D. Roman, "Norwegian state of estate: A reporting service for the state-owned properties in Norway," in *Proc. 9th Int. Web Rule Symp. (RuleML)*, in *Lecture Notes in Computer Science*, vol. 9202, N. Bassiliades, G. Gottlob, F. Sadri, A. Paschke, and D. Roman, Eds. Berlin, Germany: Springer, Aug. 2015, pp. 456–464.
- [33] L. Shi, N. Nikolov, D. Sukhobok, T. Tarasovac, and D. Roman, "The prodatamarket ontology for publishing and integrating crossdomain real property data," in *Territorio Italia. Land Administration, Cadastre and Real Estate*, no. 2. Agenzia delle Entrate, 2017, pp. 9–35.
- [34] L. Shi, D. Sukhobok, N. Nikolov, and D. Roman, "Norwegian state of estate report as linked open data," in *Proc. Move Meaningful Internet Syst. (OTM) Conf.*, in *Lecture Notes in Computer Science*, vol. 10574, Rhodes, Greece: Springer, 2017, pp. 445–462.
- [35] J. Brooke, "SUS—A quick and dirty usability scale," in *Usability Evaluation in Industry*, P. W. Jordan, B. Thomas, I. L. McClelland, and B. Weerdmeester, Eds. London, U.K.: Taylor & Francis, 1996.
- [36] J. Sauro and J. R. Lewis, *Quantifying the User Experience: Practical Statistics for User Research*. Amsterdam, The Netherlands: Morgan Kaufmann, 2012.



GUILLELMO VEGA-GORGOJO is currently an Associate Professor with the University of Valladolid, Spain, and a former member of the Analytical Solutions and Reasoning Group, University of Oslo, Norway. His research interests include visual user interfaces and system architectures for the semantic web. He has authored more than 70 scientific publications.



LAURA SLAUGHTER holds an Adjunct Associate Professorship with the Informatics Department, University of Oslo. She works primarily in the area of healthcare informatics with a focus on the reuse of health information system data for medical decision-making and personalized medicine. She is active in the areas of knowledge representation and language technologies: ontology and scalable data access, data exploration, and capturing knowledge from texts.



BJØRN MARIUS VON ZERNICHOW received the master's degree in informatics from the University of Oslo, Norway. He is currently pursuing the M.Sc. degree with SINTEF AS, Norway. He has experience in research and software development related to data management, data integration, data enrichment, and big data. He has been involved in carrying out applied research projects related to data-driven innovation with knowledge graphs.



NIKOLAY NIKOLOV received the joint Erasmus Mundus M.Sc. degree in service engineering from Stuttgart University, University of Crete, and Tilburg University. He is currently a Research Scientist with SINTEF. He has experience in both research and software development related to data management, data integration, data enrichment, big data, and the semantic web. Over the past years, he has been involved in planning and carrying out applied research projects related to data-driven innovation with knowledge graphs.



DUMITRU ROMAN currently works as a Senior Research Scientist with SINTEF AS, Norway. He has wide experience with initiating, leading, and carrying out (research-intensive) projects on data management and service-oriented topics. He is currently active in the data management field, particularly in the emerging data-as-a-service (DaaS) domain. He holds an Adjunct Associate Professorship at the University of Oslo, Norway.

...