Visual Querying LOD sources with LODeX

Fabio Benedetti
Università di Modena e Reggio Emilia - Dipartimento di Ingegneria "Enzo Ferrari"
Via Pietro Vivarelli 10
Modena, Italy
fabio.benedetti@unimore.it

Sonia Bergamaschi
Università di Modena e Reggio Emilia - Dipartimento di Ingegneria "Enzo Ferrari"
Via Pietro Vivarelli 10
Modena, Italy
sonia.bergamaschi@unimore.it

Laura Po
Università di Modena e Reggio Emilia - Dipartimento di Ingegneria "Enzo Ferrari"
Via Pietro Vivarelli 10
Modena, Italy
laura.po@unimore.it

ABSTRACT

The Linked Open Data (LOD) Cloud has more than tripled its sources in just three years (from 295 sources in 2011 to 1014 in 2014). While the LOD data are being produced at a increasing rate, LOD tools lack in producing an high level representation of datasets and in supporting users in the exploration and querying of a source. To overcome the above problems and significantly increase the number of consumers of LOD data, we devised a new method and a tool, called LODeX, that promotes the understanding, navigation and querying of LOD sources both for experts and for beginners. It also provides a standardized and homogeneous summary of LOD sources and supports user in the creation of visual queries on previously unknown datasets.

We have extensively evaluated the portability and usability of the tool. LODeX have been tested on the entire set of datasets available at Data Hub[1], i.e. 302 sources. In this paper, we showcase the usability evaluation of the different features of the tool (the Schema Summary representation and the visual query building) obtained on 27 users (comprising both Semantic Web experts and beginners).

Keywords

LOD, Schema Extraction, Schema Summarization, Visual Query Generation, SPARQL Query Generation

1. INTRODUCTION

It has been eight years since Tim Berners-Lee designed the Linked Data Principles. Now the Web of Data consists of more than a thousand of datasets collecting several billion of triples[2]. The LOD dataset generation is also encouraged by the Open Access trends and its importance has been highlighted by the report[3] produced by the Open Data Barometer of the 2014: "In 2014 the G20 largest industrial economies followed up by pledging to advance open data as a tool against corruption, and the UN recognized the need for a Data Revolution to achieve global development goals". Although, the LOD cloud is growing more and more, navigation and visualization of Linked Data is still at the beginning.

Several portals, such as the well known Data Hub, catalog datasets that are available as LOD on the Web and provide keywords search methods to identify a dataset of interest. Usually, a user have to manually explore a new dataset using SPARQL queries to understand if the dataset really contains the information that he is looking for. It follows that a user with no SPARQL knowledge cannot become a consumer of the data contained in the LOD Cloud. Even for a skilled user this is not a easy task because there are no fixed modeling rules in the design of the structure a LOD dataset; usually, external classes and properties are used within a dataset without formally define how they are related to the classes defined locally. Moreover, a great number of datasets is published without a real documentation that could help on revealing their structure.

Our tool, LODeX, aims to solve the above issues in order to empower users without technical skills in exploring, understanding and extracting knowledge from a LOD dataset without any a-priori knowledge on the source itself. In particular it aims to provide: (1) an high level Schema Summary able to capture structural information of a LOD source, to enable classes and properties browsing; (2) a powerful and intuitive visual query builder, to empower users in the in-dept exploration of the instances of the source and eventually to generate a SPARQL query able to extract the piece of knowledge to which the user is concerned. The tool takes advantage of a query refinement panel and a SPARQL compiler that capture each change in the visual query and refreshes of the corresponding SPARQL query and its result.

In this paper, we describe LODeX and we test the portability of the tool on more than 300 datasets to demonstrate that our tool can be used with the great part of the datasets belonging the LOD cloud. Moreover, we conducted a usability evaluation in order to show the effectiveness of LODeX in representing the structure of a dataset and in supporting the user in building queries on an unknown LOD source. The results demonstrate the effectiveness of the tool and, further, highlight future lines of development.

The remainder of the paper is structured as follows. We discuss related work in Section 2. We draw the architecture and a motivation example in Section 3. Section 4 illustrates a use case scenario, while Section 5 reports the evaluation on portability and usability of the tool. Finally, Section 6 sketches the conclusion and the future lines of extension for LODeX.

[1] www.datahub.io

[2] http://linkeddatacatalog.dws.informatik.uni-mannheim.de/state/

[3] http://barometer.opendataresearch.org/report/summary/

K-CAP 2015 October 07-10, 2015, Palisades, NY, USA
ISBN 978-1-4503-3849-3/15/10Ě$15.00
DOI: http://dx.doi.org/10.1145/2815833.28158492015 ACM.

This work has been accomplished in the framework of a PhD program organized by the Global Grant Spinner 2013 and funded by the European Social Fund and the Emilia Romagna Region.

	Availability Online	Scale: Whole Dataset	Scale: Instance Level	Querying: Visual Building	Querying: SPARQL Generation/Edit	Querying: By Keyword	Result Visualization
LOD Visualization	✓	✓					List and adv. visualization
ProLOD	∗	✓					Adv. visualization
LODlive	∗		✓			✓	Graph visualization
LODmilla	✓		✓			✓	Graph visualization
gFacet	∗		✓	✓		✓	Graph visualization
iSPARQL	✓		✓		✓		List and adv. visualization
SPARKLIS	✓		✓			✓	List
LD Query Wizard	✓		✓	✓		✓	List and adv. visualization
LODeX	✓	✓	✓	✓	✓		List

Table 1: Visualization, exploration and query tools (∗ it is provided an online demo)

2. RELATED WORK

Several researchers have attempted to support users in LOD source visualization, browsing and in the definition of complex queries allowing fancy visualization of the results. Table 1 contains a comparison of different tools based on visualization and querying features[4].

As shown in the table, we can distinguish between two major groups: the tools that focus on providing an overall overview of the whole structure of the datasets and the tools that provide just an instance level view of the datasets and supply query functionalities.

In the first group, we find LOD Visualization and ProLOD; tools that aim to provide to users an high level analysis of a LOD dataset. In particular, LOD Visualization is a prototype based on the Linked Data Visualization Model [8], and it allows to build analysis, transformations and visualizations of Linked Data. ProLOD [1] automatically provides a group of statistical analysis regarding the content of a dataset, but it does not foresee any querying possibility.

The second group of tools are able to provide visual querying functionalities and advanced visualizations for the query results, but their focus is limited to the instance level. LD Query Wizard [13] allows to visualize an instance selected through keyword search and it uses a powerful tabular view that permit users to explore the neighborhood of this instance. LODlive and LODmilla [14] provide a visually appealing way to explore information associated with an instance using a graph visualization. Also, gFacet [11, 12] uses the same strategy of exploration (with a graph visualization), but in this case, each node is a class that contains a list of instances and the user can link new nodes (classes) as if he/she was building a visual query. SPARKLIS [10] implements a fascinating approach in which a SPARQL query is composed as if the user was composing a natural language request to the dataset. ISPARQL [15] allows to incrementally build a SPARQL query by extending it step by step; the main issues of this approach are that the user is required to have a good knowledge of the Semantic Web technologies and to understand the schema of the LOD source for defining a SPARQL query that retrieve interesting information.

As reported in [9], the majority of the tools for data visualization requires the user to manually explore the dataset and they are not able to provide a synthetic *schema* of the data contained in a single source. LODeX differs from the tools described above since it provides a synthetic representation of LOD source schema and the user can use it to build visual queries. However, LODeX has some limitations: it is not able to perform keyword queries, moreover there are large areas of improvement in the result visualization of the query.

[4]Among the variety of tools that handle Linked Data, we selected those able to connect to SPARQL endpoints. The comparison reported in table 1 is not intended to be exhaustive.

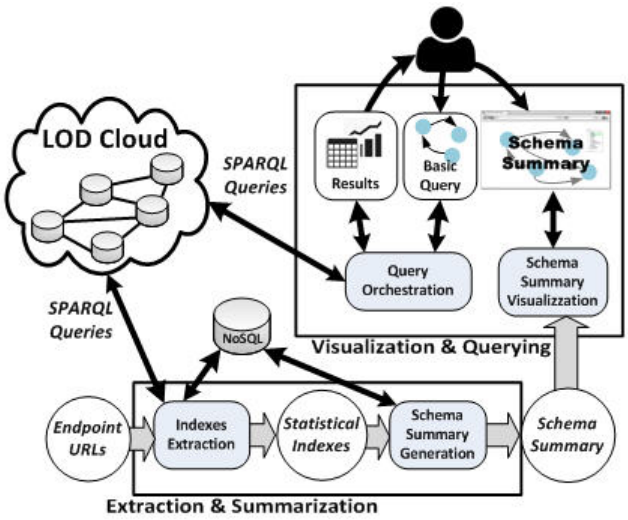


Figure 1: LODeX Architecture

3. ARCHITECTURAL OVERVIEW

LODeX consists of four distinct components, each responsible for a specific activity, named: (1) Indexes Extraction, (2) Schema Summary Generation, (3) Schema Summary Visualization, (4) Query Orchestration.

The components interact in order to: produce a visual Schema Summary (i.e. a high-level representation of the LOD source); provide it to the users; translate the visual query that a user might compose in a SPARQL query and to retrieve the results. The interaction is illustrated in Figure 1. For an easy reuse, all the contents extracted and processed are stored in MongoDB, a NoSQL document database (since it allows a flexible representation of the indexes).

3.1 Indexes Extraction

In a RDF graph the RDFS/OWL triples used to define a vocabulary or an ontology describe the intensional knowledge,while the instances and their datatype and object properties compose the extensional knowledge. In Figure 2 an example of the RDF graph representing a LOD source is displayed. The intensional knowledge is conveyed in the triples shown on the top of the figure, while, on the bottom, we have triples that describe three instances and compose the extensional knowledge.

The extraction process takes as input the URL of a SPARQL end-

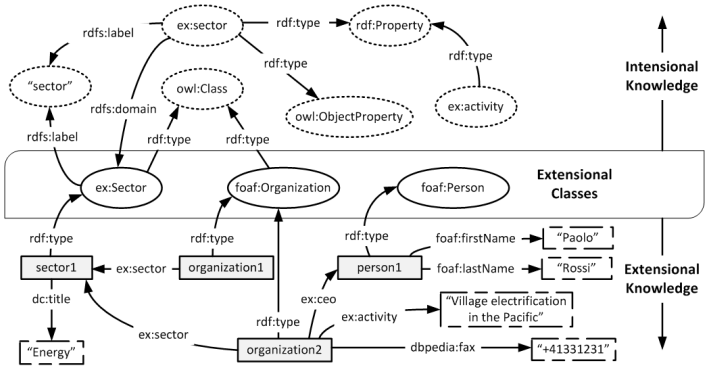


Figure 2: An example of the RDF Graph partitioning between intensional and extensional knowledge.

point and generates a set of queries able to extract a set of indexes from the extensional knowledge (extensional group of Statistical Indexes in [3]). These indexes are composed by sets of couple (c,p) where c is a class and p is a property:

- **SC** (Subject Class) contains object properties p and their domain class c.
- **SCl** (Subject Class to literal) contains datatype properties p and their domain class c.
- **OC** (Object Class) contains object property p and their range class c.

The IE process also inspects the number of times each index appears in a dataset; these information are stored together with each index since they are used to generate the Schema Summary. Table 2 lists the indexes extracted from the extensional knowledge of the example in Figure 2.

Name	Values
Classes	{ ex:Sector, foaf:Person, foaf:Organization}
SC	{(foaf:Organization,ex:ceo), (foaf:Organization,ex:sector) }
SCl	{ (foaf:Person,foaf:firstName), (foaf:Person,foaf:lastName), (foaf:Organization,ex:dbpedia:fax), (ex:Sector,dc:title), (foaf:Organization,ex:activity), (foaf:Organization,dbpedia:fax) }
OC	{ (ex:Sector,ex:sector) }

Table 2: Classes and indexes extracted from the extensional knowledge of the source depicted in Figure 2

3.2 Schema Summary Generation

The Schema Summary (SS) of LOD source is created by exploiting information contained in the indexes described in the previous Section. The number of instances of each class and the number of times a index appear in a dataset are exploited in order to discover how the classes are connected in the extensional knowledge; thus, the SS is the schema of a dataset inferred from the distribution of the its instances[5].

[5]Major detail of the LODeX approach can be found here:http://dbgroup.unimo.it/lodex_model/lodex

DEFINITION 1 (SCHEMA SUMMARY (SS)). *A Schema Summary S, derived from a RDF dataset, is a pseudograph: S = <C, P, s, o, A, m,Σ_l, l, count>, where:*

- *C contains a set of c, where c is a Class of the RDF dataset. The elements of C represent the node of the pseudograph.*
- *P contains the properties between Classes of the RDF dataset. The elements of P represent the edges of the pseudograph.*
- *s: $P \rightarrow C$ is a function that assigns to each property $p \in P$ its source class $c \in C$.*
- *o: $P \rightarrow C$ is a function that assigns to each property $p \in P$ its object class $c \in C$.*
- *A contains the attributes of Classes of the RDF dataset.*
- *m: $A \rightarrow C$ is a function that maps each attribute $a \in A$ to the class $c \in C$ to which it refers.*
- *Σ_l is the finite alphabet of the available labels.*
- *l: $(C \cup P \cup A) \rightarrow \Sigma_l$ is a function that assigns to each class, property or attribute its label.*
- *count: $(C \cup P \cup A) \rightarrow \mathbb{N}$ is a function that assigns to each property or attribute the number of times it appears in the RDF dataset, and to each class the number of instances of the class itself.*

An attribute $a \in A$ represents the existence of a datatype property with domain the class $c \in C : m(a) = c$, while a property $p \in P$ represents the existence of an object property p with domain $c_1 \in C$ and range $c_2 \in C : s(p) = c_1 \wedge o(p) = c_2$. In Figure 3 a representation of the SS of the previous example (shown in Figure 2) is depicted.

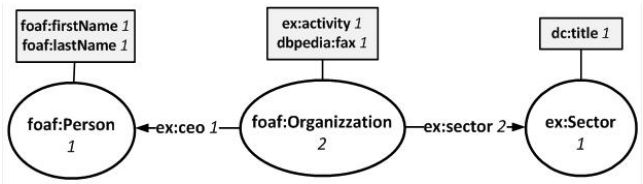


Figure 3: The SS of the LOD source represented in Figure 2. The white circles represents classes (C), while the attributes (A) are shown in the gray boxes. Finally, the edges describe the properties (P). Each element is equipped with a numerical value representing the number of occurrences/number of instances.

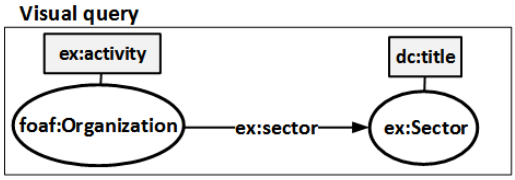
This kind of formal definition brings several advantages: the SS can be easily be stored and retrieved from MongoDB, storing the SS in a triplestore would have involved well known performance issues that would lead to worsening the performance of LODeX; the SS can be directly visualized in the GUI of LODeX and it makes possible the query building feature.

3.3 Schema Summary Visualization

The visualization is performed by a web application through which the user can interact for browsing the SS. The web server is implemented in Python, while the user interface uses different Javascript libraries to produce an interactive web application. In particular, we used Polymer to manage the GUI, a new library that allow to design applications according to the Material Design principles using Web Components[6]. We used Data Driven Documents[7]

[6]http://www.w3.org/standards/techs/components

[7]http://d3js.org/



SPARQL query

```
SELECT ?Organization ?activities ?Sector ?title
WHERE {
?Organization a <http://xmlns.com/foaf/0.1/Organization>.
 ?Organization <http://reegle.info/schema#activities>
       ?activities.
 ?Sector a <http://reegle.info/schema#Sector>.
 ?Sector <http://purl.org/dc/elements/1.1/title> ?title.
 ?Organization <http://reegle.info/schema#sector> ?Sector.
}
```

Figure 4: An example of a visual query created on the Schema Summary shown in Figure 3 and its translation in SPARQL.

to create the interactive Schema Summary, and Sgvizler[8] to allow the querying of the remote endpoints and to display the results. The visualization of the Schema Summary has been also presented in the demo [4].

3.4 Query Orchestration

The Query Orchestrator manages the interaction between the GUI and the user in composing the visual query, in the generation of the SPARQL queries and in the submission of it to the remote endpoint.

The classes, properties and attributes selected by the user in the visual query participate to the composition of a basic query[9] (Q). Q has a tree structure that overlaps the SS graph, the nodes of the tree are classes $\in C$, while the leafs can be both classes $\in C$ or attributes $\in A$. Graphically, a user starts composing a basic query by selecting the first class in the SS, then, if the user selects a property for this first class, also the connected class is shown in the query panel and the edge and vertex are added to the tree. The user may also select the attributes of each class: in this case, the tree is further enriched with edges and leafs.

The Query Orchestrator translates the basic query into a SPARQL query through a compiler. The compiler exploits an iterative algorithm that traverses the basic query tree to produce the SPARQL query. The Query Orchestrator is able to compile non cyclic SPARQL query of any length; it allows the use of these SPARQL operators: AND (.), OPTIONAL (also nested), FILTER, ORDER BY, OFFSET and LIMIT.

The example introduced in Figure 4 shows a simple query built on the Schema Summary of Figure 3. This query has been composed by selecting the class *foaf:Organization*, its attribute *ex:activity* and the property *ex:sector*. The selection of this property automatically results in the selection of the object class *ex:Sector*, then, we also add the attribute *dc:title*. From this graphical query, the Query Orchestrator module generates the SPARQL query shown in the bottom part of the Figure 4.

4. A USE CASE SCENARIO

Here, we refer to a hypothetical use-case involving a company in the clean energy sector. The company has its own products and services and attempts to discover new information on renewable energy in the country where it is located. It is very likely that looking on portals like Datahub, the company detects the “Linked Clean Energy Data” dataset[10]. This dataset, composed of 60140 triples, is described as a “Comprehensive set of linked clean energy data”. By using LODeX, the structure of the dataset is revealed and it can be easily browsed (see Figure 5)[11]. At a glance, the user can have the intuition of all the main classes (the nodes in the graph) and the connections among them (the arcs), besides the number of instances defined for each class (reflected in the dimension of the node). Focusing on the color of the nodes, a user can understand which classes are defined by the provider of the source and which others are taken from external vocabularies (in this case we can see that some of the class definitions are acquired from Foaf, Geonames.org and Skos) using the legend (Fig 5 Sect A). By positioning the mouse on a node, more information about the class is shown.

As depicted in Figure 5 (Sect B), the source collects 1869 organizations and each organization is described by some attributes (Sect D) together with the average number of times each attribute is used by an instance of the class, for example not all the instances have a zip code (0.88), whereas all of them have more than one name (1.60). Moreover, a class is linked to others by some properties (Sect C). By navigating the schema, a user might also discover that each organization is link to roughly 3 sectors, but then each sector (36 sectors in total) is linked to 151 organizations.

The user has to select a root node to start building a visual query (”Organization" in Figure 5). Now the user can add some attributes to the current class by clicking the buttons on the left of the attributes name (M: mandatory, O: optional). In Figure 5 the user select 3 optional attributes (”name", ”abbreviation" and ”street") for the class ”Organization". The user can also add other classes linked to the current class through a specific property by clicking the button on the left of a property in the property panel (Sect C). In Figure 5 the user added 2 mandatory classes/properties (”activeIn" ”Feature" and ”sector" "Sector"). The user can look at the visual query that he is building (Fig. 5 Sect E) and use it in order to focus on the different components of the query and add other attributes or properties/classes. At this point, the user can generate the query clicking the ”Generate" button which brings the user in the query refinement panel (Figure 7).

In the refinement panel (Fig. 7) the user can visualize the SPARQL query (F) that has been generated and he may manually modify it. He can also choose to visualize directly the result of the query by selecting the result tab or enable the automatic compiler (E) and modify the query by using the interface on the top (A,B,C,D) visualizing the results that change according to his refinement. After any change the query is compiled and automatically sent to the endpoint. In particular, the user can: (A) add or remove filter condition on the attributes contained in the query; (B) modify the optionality of attributes/classes or remove one of them from the query; (C) remove the pagination of the results, or modify the page size; (D) insert or remove ordering condition.

5. EVALUATION

We propose three kinds of evaluations regarding LODeX: first, we analyze the portability of the LODeX approach; then, we evaluate the level of expressiveness of the SPARQL queries that can be generated by LODeX; finally, we provide the result of a usability evaluation performed with anonymous users. A deep evaluation of the performance of IE process can be found in [3].

[8] http://dev.data2000.no/sgvizler/

[9] the formal definition of Q is out of scope of this paper and can be found at: http://dbgroup.unimo.it/lodex_model/lodex#x1-70003.1

[10] http://data.reegle.info/

[11] The visual summary of this source is also available at http://dbgroup.unimo.it/lodexCleanEnergy

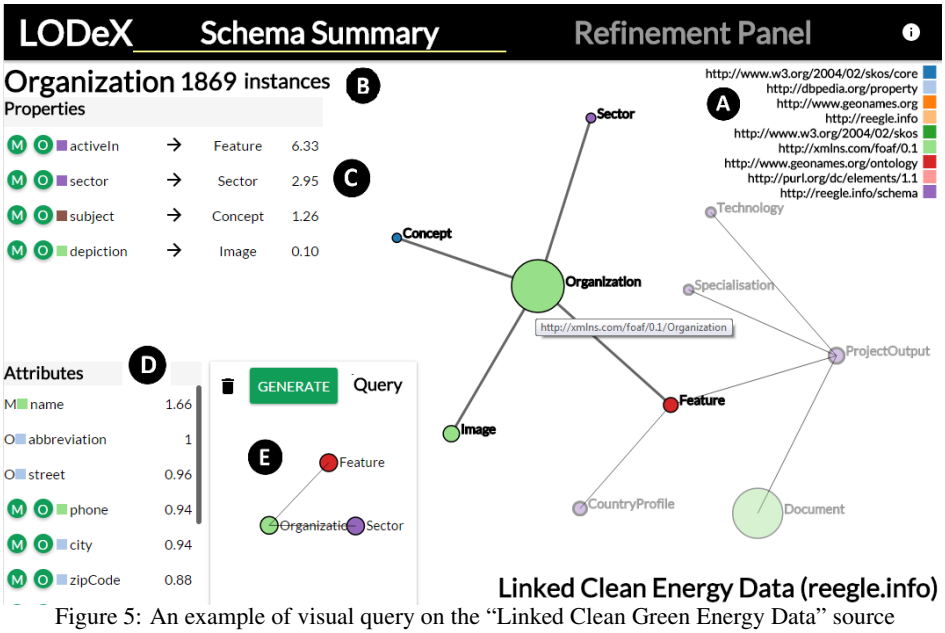


Figure 5: An example of visual query on the "Linked Clean Green Energy Data" source

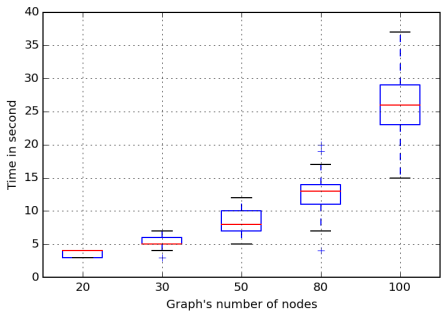


Figure 6: Distribution of the micro-tasks execution time grouped for graph size.

Test	Nov 2014
Reachable datasets	302
SPARQL 1.1 compatible	206
Extraction completed	185

Table 3: Number of Dataset used perform the portability evaluation

5.1 Portability to SPARQL endpoints

LODeX has been designed to be a tool able to work with each dataset provided of a SPARQL endpoint. Thus, we use the complete list of SPARQL endpoints contained in DataHub as test set.

Table 3 reports the number of datasets that were examined; 302 datasets were online when we performed the test. The IE process use a subset of SPARQL operator to extract the indexes, so, just 206 datasets were compatible. Another well known issue is the bad performance of some SPARQL endpoint, for this reason the number of endpoint for which we was able to generate the SS decrease to 185, that remains a good result because we obtained a SS from the 61% of the reachable endpoints.

Now, we extend this portability evaluation to the GUI of LODeX and we inspect two aspects: success/failure of the communication with the endpoints; clarity of the graph representation of the SS[12].

[12]the results of report can be consulted online at http://dbgroup.unimo.it/detasetsLodexPortability.html

We executed preliminary usability test in our laboratory using 5 students to find out how many the size of the graph affects its clarity. We asked the students to individuate a specific node in graphs of different size (20, 30, 50, 80 and 100 nodes) and we measured the time taken for each task. We provided to students 25 tasks each (5 tasks for each graph size). The results are shown in the Figure 6, as you can see the finding time increases almost linearly when the dimension of the graph is less than 80 nodes. For this reason, we decided to not consider the datasets having more than 80 nodes. The number of these datasets is 40 and they represent the 21% of the total. Possible solutions to this issue will be discussed in as future work in Section 6.

Out of the remaining 145 endpoints, 7 were not online when the test was performed, 28 returned to the user interface a non-standard response. The LODeX web application makes an AJAX request to the endpoint containing the query and requiring a response encoded through *JSONP (JSON with padding)*. Since some endpoints (28 in our case) were not able to encode a JSONP response, they replied with a non-standard response. Finally, 110 endpoints, almost the 60% of the total[13], successfully pass the test.

5.2 SPARQL expressiveness

To evaluate the level of expressiveness of the queries generated we inspected how many of the queries composing the BSBM benchmark [6] (Berlin SPARQL Benchmark) could be generated using LODeX. These set of queries is formed by queries usually used to explore a new dataset. LODeX would be able to generate 6 of 10 queries proposed by the benchmark, a good result taking in consideration that a user without any knowledge of SPARQL could be able to generate them with LODeX. The four excluded queries contain SPARQL operator not supported by our tool: UNION, CONSTRUCT and DESCRIBE. LODeX is able to generate all the queries involving any type of JOIN and FILTER operation except for the cyclic queries. Indeed, the SPARQL compiler is able to automatically translate a basic query, the structure of which is a tree.

5.3 Usability Evaluation

[13]You can browse these datasets using the demo of LODeX available at: http://dbgroup.unimo.it/lodex2

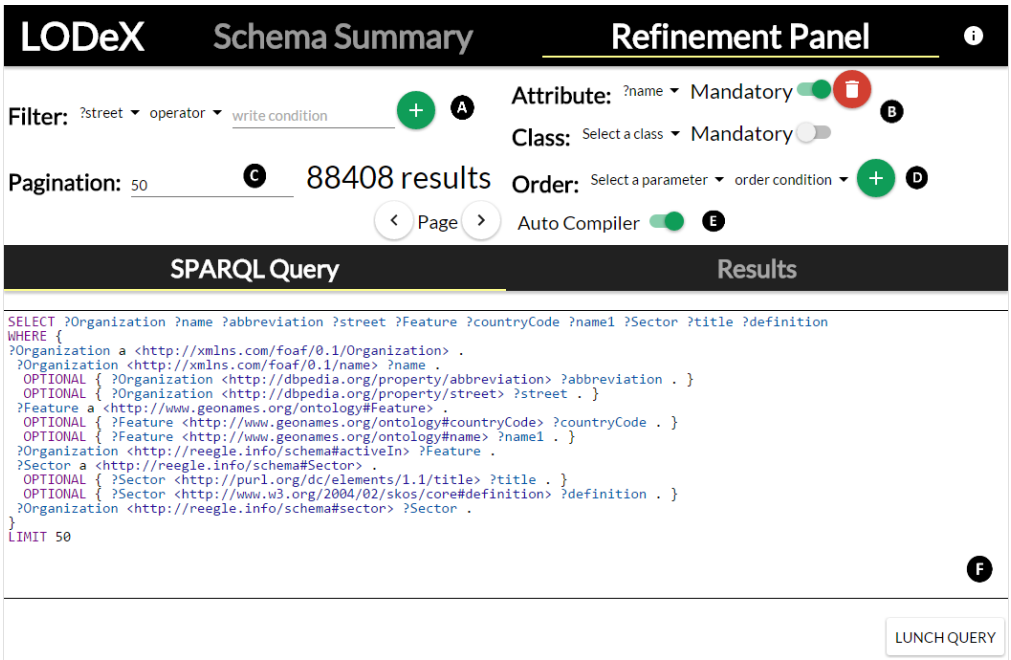


Figure 7: An example of the translation of the visual query of Figure 5 into the corresponding SPARQL query.

This section summarizes the results of an evaluation performed as an online survey[14] compiled by anonymous users. Among the users involved, 22 were enrolled from IT communities and others 5 were bachelor students. We divided the survey in two distinct parts: the first aims to verify if the graph visualization of the SS is clear in representing the structure of a dataset; the second part intends to prove if the visual query panel is a powerful and adequate way for generating SPARQL queries. The survey collects the results of a sparse set of users aged between 23 and 43 years (Fig. 8) with different Semantic Web technologies skills (as shown in Figure 9). This is an ideal scenario to prove the effectiveness of the tool on users with different background knowledge.We used 3 different datasets in the survey: (D1) Bio2RDF - INOH - pathway database of model organisms[15]; (D2) Linked Open Aalto Data Service - Open data published by Aalto University[16]; (D3) Nobel Prizes - Linked Open Data about every Nobel Prize[17].

5.3.1 *Methodology*

The survey encloses a short tutorial containing a description of the SS and a short video where the functionalities of query generation are explained[18]. Each of the two parts is composed by microtasks designed to evaluate the effectiveness of LODeX in addressing its two main goals.

Schema Summary Browsing Functionality - We propose two anonymous SS generated from two datasets (D1 and D2). The tasks that we asked the users to perform are listed in Table 4 (T1 to T4).

Query Generation Functionality - We asked to users to generate 4 different queries from natural language requests (the requests are listed in Table 4 from Q1 to Q4).

Finally, we asked to compile a SUS [7] questionnaire and reply to a usability questionnaire. In particular, we asked to score, on a scale of 1-5, the following sentences: “I found the Schema Summary was easy to browse””; It permits to have an overview about the structure of a Dataset”; “The visualization of the Schema Summary is clear”. For the second part, we asked questions regarding the SPARQL query generation feature and the overall tool: "How do you evaluate your knowledge about SPARQL?"; "If you have already written SPARQL queries, how do you find using LODeX compared to manually writing SPARQL queries?"; "Any comments? What was good / bad / unexpected / difficult?".

T1:	Find out the topic of each dataset	D1,D2
T2:	Find out the class with the largest number of instances	D1,D2
T3:	Find out the classes connected to a given class chosen by us	D2
T4:	Find out the most used attribute of a class chosen by us	D2
Q1:	Return all the different category of Nobel prizes	D3
Q2:	Return a table containing the list of winners of a Nobel prizes ordered by the name of the winner; the table has to contain the date of birth of the winner.	D3
Q3:	Find the award files related to the award of Peter W. Higgs	D3
Q4:	Find the organizations that won a Nobel prize after the 1999	D3

Table 4: Tasks and queries used in the LODeX evaluation and the corresponding datasets.

5.3.2 *Quantitative evaluation*

We evaluate the correctness of the answers provided by users for the tasks listed in Table 4.

Schema Summary Browsing Functionality - The tasks belonging to this section obtain an accuracy of the 91% (Table 5). We asked to complete these tasks without querying the dataset, but just brows-

[14]The survey can be compiled at this url:http://goo.gl/forms/FRSRWKLSq4
[15]http://datahub.io/dataset/bio2rdf-inoh
[16]http://datahub.io/dataset/linked-open-aalto-data-service
[17]http://datahub.io/dataset/nobelprizes
[18]The tutorial is accessible at http://dbgroup.unimo.it/LODeXGuide.html

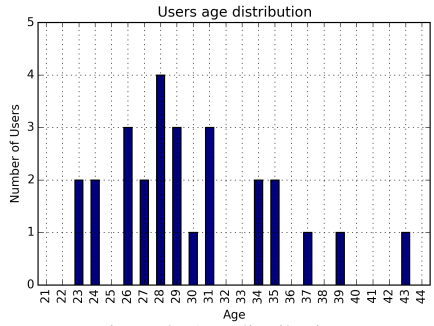


Figure 8: Age distribution.

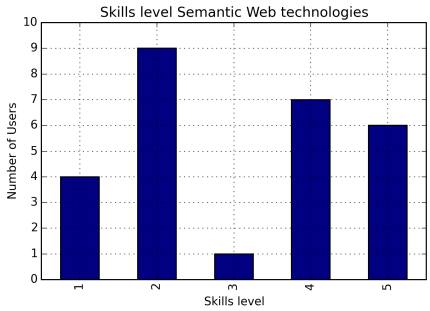


Figure 9: Semantic Web skill distribution.

ing the Schema Summary, so we obtained a high accuracy. The students that completed the survey in our laboratory were able to complete these task in less than 5 minute in average.

Query Generation Functionality - These group of tasks obtained an overall accuracy of 90%. This is a very good result because the last 3 queries are quite complex; in fact, they involve 2 or more classes with a filter or an order condition . The students that completed the survey in our laboratory were able to complete these task in less than 15 minute in average. This is a very promising result, in fact, all the students enrolled had a very low knowledge of SPARQL.

5.3.3 Qualitative evaluation

We evaluate the SUS score obtained and the answers to the qualitative question proposed.

Schema Summary Browsing Functionality - In Figure 11 you can have an overview of the SUS score obtained by the 27 users, the results are clustered according to the knowledge of the user of the Semantic Web technologies. The SUS overall median value is 85 and, according to [2], we can classify this functionality to *Excellent*. The median values obtained distinguishing among skilled and unskilled user are rather similar (82.5 vs 87.5), so we can assume that this functionality has been appreciated by both kind of users. Moreover, we also request to rank the level of agreement to three sentences regarding the SS (see Figure 10 for the distribution of the results) and practically most users think that: it is easy to browse;

Task	Number	n Correct	% Correct
T1	54	48	89%
T2	54	48	89%
T3	27	23	89%
T4	27	27	100%
Total	**162**	**148**	**91%**

Table 5: Results of the tasks listed in Table 4

Task	Number	n Correct	% Correct
Q1	27	27	100%
Q2	27	26	96%
Q3	27	22	81%
Q4	27	23	85%
Total	**108**	**98**	**90%**

Table 6: Results of the queries listed in Table 4

it can work as documentation of a dataset; its visualization is clear.

Query generation functionality - This functionality uses all the features of the tool, so we can assume that the SUS scores obtained in this step represent the global SUS score of LODeX. Therefore, the distribution of the SUS score obtained for LODeX is shown in Figure 12 and we obtained a median SUS score of 82.5 that classifies LODeX as *Excellent*, always according to [2]. Also in this case, we do not find particular differences among the median value of the score among skilled and unskilled users (82.5 vs 85). The fact that, both skilled and unskilled users equally appreciated LODeX, according to the SUS scores, demonstrates that the final user can be unaware to Semantic Web technologies to explore and query LOD sources with LODeX. That was one of the main goal of LODeX in order to increase the usage of LOD sources. Users who did not know SPARQL were able to query a dataset LOD for the first time; an user answers to this question, "If you have already written SPARQL queries, how do you find using LODeX 2.0 compared to manually writing SPARQL queries?", like this: "Just written my first SPARQL queries using LODeX. Nice". On the other hand, one skilled user answers to the question above in this way: "LODeX is cognitively less demanding". We received also some criticisms concerning some aspects of the GUI (e.g. browser rendering differences) that will be very useful for improving LODeX. Another criticism regards the graph visualization of the SS that can became complex for huge dataset and starting a query can be difficult for a user.

6. CONCLUSION & FUTURE WORKS

In this paper, we presented LODeX, a tool for visual exploration and querying of LOD sources. LODeX unveils the intrinsic structure of a LOD source by providing a summarized view of the dataset and allow users to visually compose/refine a query addressed to this source.

LODeX has proven to be an effective tool in facilitating users' interaction with LOD sources. Moreover, writing SPARQL queries can be a time-consuming and boring task also for experts, thus, navigating the inferred schema of a dataset and selecting classes and attributes of interest can strongly simplify the formulation of a query, making more pleasant the consumption of Linked Data. Portability tests showed that LODeX is able to process 61% of the accessible SPARQL endpoints and to render 59% of the LOD sources.The survey, conducted on 27 users, has revealed a good level of usability with a SUS classification as "Excellent". A complete demo of the tool has been also presented in the demo [5]

However, some limitations arise from the evaluation of the tool. First of all the graph visualization of the SS can become messy for huge dataset. This might affects the portability of LODeX, therefore we are currently studying different solutions to solve this drawback: for example to apply clustering techniques and group together some sets of nodes with similar characteristics or limit the number of nodes visualized to the neighborhood of the node that is the current focus of the user. The first solution allows to visualize the structure of the whole dataset, but the query building func-

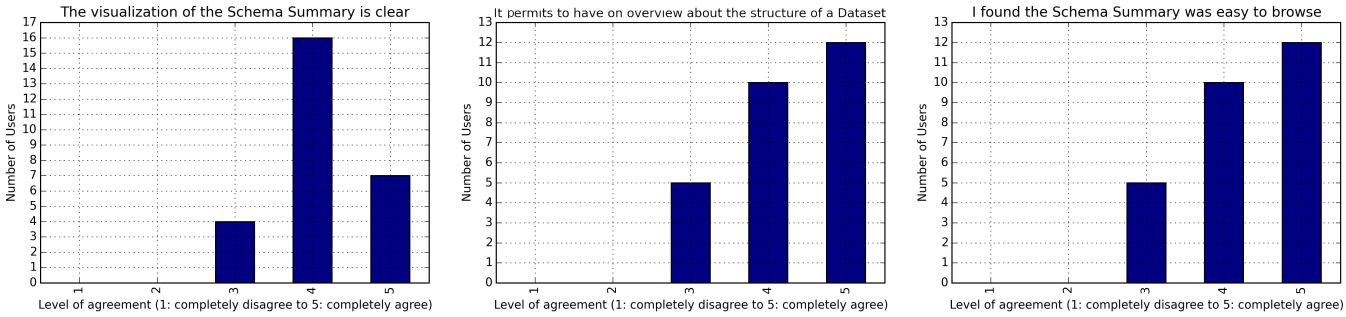


Figure 10: Distribution of the evaluations by users about the usability of the SS browsing functionality.

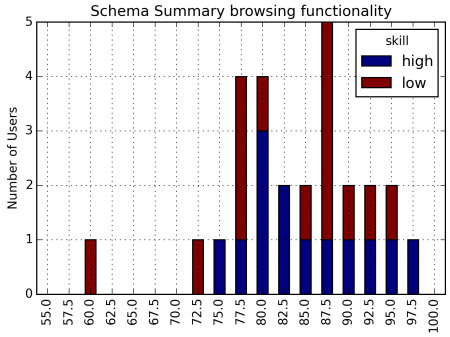


Figure 11: Distribution of SUS score for the Schema Summary browsing.

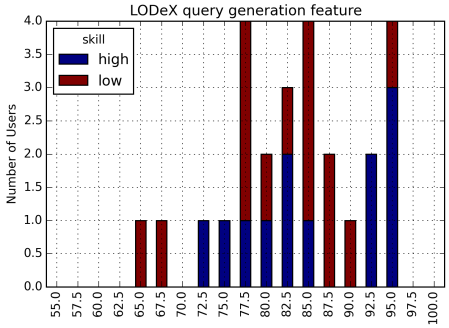


Figure 12: Distribution of SUS score for the query generation functionality.

tionality might be affected. With the second option, we does not affect the query building, but we lose the possibility to represent the whole dataset. Moreover, the use of keyword search techniques could significantly improve the selection of elements of a visual query.

7. REFERENCES

[1] Z. Abedjan, T. Grütze, A. Jentzsch, and F. Naumann. Profiling and mining RDF data with prolod++. In I. F. Cruz, E. Ferrari, Y. Tao, E. Bertino, and G. Trajcevski, editors, *IEEE 30th International Conference on Data Engineering, Chicago, ICDE 2014, IL, USA, March 31 - April 4, 2014*, pages 1198–1201. IEEE, 2014.

[2] A. Bangor, P. Kortum, and J. Miller. Determining what individual sus scores mean: Adding an adjective rating scale. *Journal of usability studies*, 4(3):114–123, 2009.

[3] F. Benedetti, S. Bergamaschi, and L. Po. Online index extraction from linked open data sources. *Linked Data for Information Extraction (LD4IE) Workshop held at International Semantic Web Conference*, 2014.

[4] F. Benedetti, S. Bergamaschi, and L. Po. A visual summary for linked open data sources. *International Semantic Web Conference (Posters & Demos)*, 2014.

[5] F. Benedetti, S. Bergamaschi, and L. Po. Lodex: A tool for visual querying linked open data. To appear in International Semantic Web Conference (Posters & Demos), 2015.

[6] C. Bizer and A. Schultz. Benchmarking the performance of storage systems that expose sparql endpoints.

[7] J. Brooke. Sus-a quick and dirty usability scale. *Usability evaluation in industry*, 189(194):4–7, 1996.

[8] J. M. Brunetti, S. Auer, and R. Garca. The linked data visualization model. In *International Semantic Web Conference (Posters & Demos)*, 2012.

[9] A.-S. Dadzie and M. Rowe. Approaches to visualising linked data: A survey. *Semantic Web*, 2(2):89–124, 2011.

[10] S. Ferré. Expressive and scalable query-based faceted search over sparql endpoints. In P. Mika, T. Tudorache, A. Bernstein, C. Welty, C. Knoblock, D. Vrandei, P. Groth, N. Noy, K. Janowicz, and C. Goble, editors, *The Semantic Web ISWC 2014*, volume 8797 of *Lecture Notes in Computer Science*, pages 438–453. Springer International Publishing, 2014.

[11] P. Heim, T. Ertl, and J. Ziegler. Facet graphs: Complex semantic querying made easy. In *The Semantic Web: Research and Applications, 7th Extended Semantic Web Conference, ESWC 2010, Heraklion, Crete, Greece, May 30 - June 3, 2010, Proceedings, Part I*, pages 288–302, 2010.

[12] P. Heim, J. Ziegler, and S. Lohmann. gfacet: A browser for the web of data. In *Proceedings of the International Workshop on Interacting with Multimedia Content in the Social Semantic Web (IMC-SSW'08) Koblenz, Germany, December 3, 2008.*, 2008.

[13] P. Höfler, M. Granitzer, E. E. Veas, and C. Seifert. Linked data query wizard: A novel interface for accessing SPARQL endpoints. In *Proceedings of the Workshop on Linked Data on the Web co-located with the 23rd International World Wide Web Conference (WWW 2014), Seoul, Korea, April 8, 2014.*, 2014.

[14] C. Kiefer and A. Bernstein. Lodmilla: a linked data browser for all. In *Proceedings of the Posters and Demos Track of 10th International Conference on Semantic Systems*, SEMANTiCS2014, pages 31–34. CEUR-WS.org, 2014.

[15] C. Kiefer, A. Bernstein, and M. Stocker. The fundamentals of isparql: A virtual triple approach for similarity-based semantic web tasks. In *Proceedings of the 6th International The Semantic Web and 2Nd Asian Conference on Asian Semantic Web Conference*, ISWC'07/ASWC'07, pages 295–309, Berlin, Heidelberg, 2007. Springer-Verlag.