Exploring RDF Datasets with LDscout

Anna Goy, Diego Magro and Francesco Conforti
Dipartimento di Informatica, Università di Torino, C. Svizzera 185, Torino, Italy

Keywords: Semantic Web, Linked Data, RDF Dataset, Dataset Exploration.

Abstract: In this paper, we present the formal model underlying LDscout, a Java library enabling developers to query a dataset specifying the vocabulary (ontology) they want to use and the instances they want to query about. The model is based on the formal definition of the concepts of Exploration Task and Exploration Task Solution, and is independent from the dataset. In this paper, we present the specific implementation that enables the interaction with RDF triplestores using OWL ontologies. In order to assess our approach, we report the usage of LDscout within PRiSMHA, a Digital Humanities project aimed at enhancing the access to historical archives through Semantic Web technologies.

1 INTRODUCTION

The Linked Open Data (LOD) paradigm (Heath and Bizer, 2011) has become a profitable approach in many fields, in a twofold perspective: (a) for publishing data, to gain visibility for the large amount of information otherwise buried in proprietary repositories; (b) for using data, to leverage the information widely available in a structured and machine-readable way. The number of available datasets in the LOD cloud is constantly increasing (see https://lod-cloud.net/). Among the fields that took up LOD practices, ranging from e-government to healthcare, cultural heritage has started playing a major role (Alexiev, 2016) (Edelstein et al., 2013); see also the ArCo project in Italy: http://dati.beniculturali.it.

According to the Linked Data paradigm, data are expressed as RDF triples (W3C, 2014) and RDF datasets are accessed through SPARQL endpoints (W3C, 2013). This structure implies that, in order to query LOD datasets and to ”understand” the query results, clients – being either users or applications – need to be familiar with the query language, i.e. SPARQL, and to understand the vocabulary used in the datasets.

In Section 2 we will provide a brief survey of the approaches providing friendly User Interfaces aimed at supporting human users in overcoming the complexity of SPARQL queries and the difficulties in grasping the semantic model underlying RDF datasets. In this paper, instead, we focus on the provision of suitable APIs, aimed at offering software clients (applications) an easy and flexible way to access LOD datasets. In particular, we will present the formal model underlying LDscout, a Java library enabling developers to specify the vocabulary they can understand (or they are currently focusing on) and the instances they want to query about. The library is wrapped in a RESTful web service interface, in order to offer a language-independent interoperability.

LDscout is based on a general model, that can be used to query any dataset. The specific implementation we present here enables the interaction with RDF datasets and in order to assess its usefulness, we used it within PRiSMHA (Providing Rich Semantic Metadata for Historical Archives: di.unito.it/prismha). PRiSMHA is a Digital Humanities project aimed at experimenting with a crowdsourcing approach for the construction of ontology-based formal semantic representations of the content of historical documents (Goy et al., 2017), within the overall goal of enhancing the access to historical archives through Semantic Web technologies (Goy et al., 2015). The project is funded by Compagnia di San Paolo and Università di Torino and relies on a collaboration between Computer Science and Historical Studies departments of the same University with Fondazione Istituto Piemontese A. Gramsci, member of the Polo del ’900 foundation (www.polodel900.it). The current proof-of-concept prototype we are developing within the project in the area of cultural heritage deals with archival resources concerning the students and workers protest during the years 1968-1969 in Italy.

The paper is organized as follows. In Section 2 we briefly survey the relevant work investigating the inte-

Goy, A., Magro, D. and Conforti, F.
Exploring RDF Datasets with LDscout.
DOI: 10.5220/0006957600920100
In *Proceedings of the 10th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K 2018) - Volume 3: KMIS*, pages 92-100
ISBN: 978-989-758-330-8

raction with RDF datasets. Section 3 presents the model underlying LDscout by providing a general formal definition of *Exploration Task* (Section 3.1), its instantiation on RDF triplestores and OWL ontologies (Section 3.2), and the mechanisms for computing solutions to Exploration Tasks (Section 3.3); the section is concluded by summarizing how we used LDscout within the platform developed in the PRiSMHA project (Section 3.4). Finally, Section 4 mentions the main directions of our future work.

2 RELATED WORK

”The term Linked Data refers to a set of best practices for publishing and interlinking structured data on the Web” (Ngonga-Ngomo et al., 2014), p.1. The introduction to Linked Data by Ngonga Ngomo and colleagues contains a detailed overview of the Linked Data life-cycle and discusses the state-of-the-art of different aspects of it (namely, extraction, authoring, linking, enrichment, and quality assessment). As already mentioned in Section 1, in order to query RDF datasets available in the LOD cloud, users need to be familiar with formal languages used to query such datasets (typically SPARQL), and should be aware of the semantic model undelying the dataset, in order to be able to formulate the proper query and to ”understand” the resources and properties in the query results.

In order to face such difficulties, a number of tools have been proposed, aimed at providing simple query interfaces to non-expert users – see, for example, (Heim et al., 2010) (Russell and Smart, 2010) (Haag et al., 2015) among many others.

In this perspective, a user-friendly approach to query RDF datasets is keyword search; for example, (Ouksili et al., 2016) presents an enhanced approach to search through RDF datasets using keywords: Ouksili and colleagues’ approach is based on *patterns* used to include external knowledge in the search process in order to improve the result quality. However, the most promising approach to support users in querying RDF repositories seems to be *faceted search* (Tunkelang, 2009) (Tzitzikas et al., 2017). Facets are predicate-value pairs (e.g., *gender= female*) and faceted search can be seen as an interactive model for query formulation in which users can progressively apply filters in the form of facets to narrow down the results. There are several tools based on this approach, such as (Berners-Lee et al., 2008) (Hahn et al., 2010) (Heim et al., 2008) (Hildebrand et al., 2006) (Huynh and Karger, 2009), and more recently (Fafalios and Tzitzikas, 2013) (Navarro et al., 2015) and (Graub et al., 2016), where the authors present SemFacet, a faceted search tool enhanced by the use of ontological axioms for enriching results with implicit information. Moreover, in this track, it is worth mentioning the ontology-aware system by Hyvönen and colleagues (Kurki and Hyvönen, 2010), and the application of their approach in the cultural heritage domain (Hyvönen et al., 2005).

Another tool aimed at supporting users in LOD exploration is PepeSearch (Vega-Gorgojo et al., 2016), that provides a form-oriented user interface based on the semantic model underlying the dataset to be explored. Other similar tools are Optique-VQS (Soylu et al., 2016) and Surveyor (Vega-Gorgojo et al., 2017), a CORS-enabled SPARQL 1.1 endpoint plugin available without installation, providing users with a view of the content of the selected dataset and enabling navigation through classes and instances.

One of the most challenging issue when exploring LOD sets is the mapping between semantic models as well as between entities across different datasets. As far as semantic models alignment is concerned, the relevant research area is ontology matching – see, for instance, (Potoniec et al., 2017) – but it falls outside the scope of the present paper. As far as the mapping between entities described in different datasets is concerned, it is worth mentioning some well-known *entity linking* (or *instance matching*) tools: LIMES, a time-optimized tool for link discovery (Ngonga-Ngomo and Auer, 2011) (Ngonga-Ngomo, 2012) and SILK (Volz et al., 2009). We will see how such tools can be relevant in our approach in Section 3.4.

3 LDscout

LDscout executes an *Exploration Task* on a given dataset, retrieving all data about a given set of instances, described using a given set of terms (vocabulary). We will start by defining the notions of *Exploration Task* and *Solution to an Exploration Task* in general (Section 3.1), then we will instantiate these notions on the specific case of RDF datasets and OWL ontologies (Section 3.2). Since computing the solution to an exploration task may be often unfeasible in practice, we will then characterize specific subsets of the solutions to RDF-OWL Exploration Tasks, which can be easily computed and that still provide useful insighs on data (Section 3.2); we will then show how this sub-solutions can be actually computed by means of SPARQL queries (Section 3.3).

3.1 Formal Definition of an Exploration Task

Definition 3.1. *An Exploration Task is a triple $ET = \langle V, I, ds\rangle$, where V is a non-empty and possibly infinite set of terms, I is a non-empty and possibly infinite set of instances (where $V \cap I = \emptyset$) and ds is a dataset (i.e. a set of data elements).*

Intuitively, an exploration task represents the task of retrieving from the dataset ds all and only those data elements that describe the instances in I and that are expressed by means of the terms in V. The constraint $V \cap I = \emptyset$ states that in any exploration task there is a rigid distinction between the elements that play the role of instances about which data are searched and those that play the role of terms used to describe those instances.

In a typical setting, I is the set of entities an agent (being either a human user or a software application) is focusing on and that (s)he/it is searching data about, V is a vocabulary that the agent can understand (and that denotes the set of notions the agent is currently interested in) and ds is the dataset that is being explored.

A solution to an exploration task is defined as follows:

Definition 3.2. *The solution to an Exploration Task $ET = \langle V, I, ds\rangle$ is the largest set $SOL(ET) \subseteq ds$, such that each data element in $SOL(ET)$ is about a non-empty set of instances $INST \subseteq I$ and it describes the instances in $INST$ using only terms contained in V.*

Intuitively, a solution to an exploration task is the largest subset of the explored dataset ds that describes entities specified in I, by means of terms in V.

From the definitions above, it immediately follows that:

Proposition 3.1. *Any exploration task always has one and only one (possibly empty) solution.*

For example, let us consider the following sets:

- $V_1 = \{Woman, Man, Factory, age, works\_at, collaborates\_with\}$
- $V_2 = \{Person, knows\}$
- $I_1 = \{Mary, Alice, Bob\}$
- $I_2 = \{Betty, Bob\}$
- $ds = \{Woman(Mary), Woman(Alice), Man(John), Man(Steve), Consultant(John), Consultant(Alice), age(Mary, 45), Company(ACME), works\_at(Mary, ACME), collaborates\_with(John, Mary), collaborates\_with(Alice, Mary), collaborates\_with(Steve, John)\}$

Given the sets specified above, the solution to the exploration task $ET_1 = \langle V_1, I_1, ds\rangle$ is $SOL(ET_1) = \{Woman(Mary), Woman(Alice), age(Mary, 45), works\_at(Mary, ACME), collaborates\_with(John, Mary), collaborates\_with(Alice, Mary)\}$;

As regards the problem formulation, it is worth noting that: Not all the terms employed in ds are mentioned in V_1 (e.g. $Consultant \notin V_1$, but it is used in ds); the same holds for instances ($Steve \notin I_1$, but it is described in ds); conversely, $Factory$ belongs to V_1, but it does not belong to the language of ds and Bob is included in I_1, but it is not described in ds.

Moreover, for what it concerns the solution, it is worth noting that: $Man(John)$, $Man(Steve)$, $collaborates\_with(Steve, John) \notin SOL(ET_1)$, even though Man, $collaborates\_with \in V_1$ and this is due to the fact that $John$, $Steve \notin I_1$; $Consultant(Alice) \notin SOL(ET_1)$, even though $Alice \in I_1$ and this is due to the fact that $Consultant \notin V_1$; $works\_at(Mary, ACME)$, $collaborates\_with(John, Mary) \in SOL(ET_1)$, even though $ACME$, $John \notin I_1$, and this is due to the fact that $works\_at$, $collaborates\_with \in V_1$ and $Mary \in I_1$.

As stated in Proposition 3.1, an exploration task may have an empty solution, for instance: If $ET_2 = \langle V_2, I_1, ds\rangle$, then $SOL(ET_2) = \emptyset$ (since no term in V_2 belongs to the language of ds); if $ET_3 = \langle V_1, I_2, ds\rangle$, then $SOL(ET_3) = \emptyset$ (since no instance in I_2 is described in ds).

Among the exploration tasks, it is worth mentioning those ones in which the vocabulary is the set $\overline{V}$ of all possible terms and/or the set of instances is the set $\overline{I}$ of all possible instances. Even though such tasks do not present any formal peculiarity, they are conceptually relevant, thus we spend a few words about them.

Any exploration task of the form $ET^{\overline{V}} = \langle \overline{V}, I, ds\rangle$ (with $I \neq \overline{I}$) specifies the task of retrieving from the dataset ds all the data about the instances in I, no matter the language in which such data are expressed. For example, if $ET_1^{\overline{V}} = \langle \overline{V}, I_1, ds\rangle$ (I_1 and ds specified above), we have $SOL(ET_1^{\overline{V}}) = \{Woman(Mary), Woman(Alice), Consultant(Alice), works\_at(Mary, ACME), collaborates\_with(Mary, John), collaborates\_with(Alice, Mary)\}$

Any exploration task of the form $ET^{\overline{I}} = \langle V, \overline{I}, ds\rangle$ (with $V \neq \overline{V}$) represents the task of retrieving from the dataset ds all the data expressed in terms of the vocabulary V, no matter the instances they are about. For example, if $ET_1^{\bar{I}} = \langle V_1,\ \bar{I},\ ds\rangle$ (V_1 and ds specified above), we have $SOL(ET_1^{\bar{I}}) = \{Woman(Mary),\ Woman(Alice),\ Man(John),$ $Man(Steve),\ works\_at(Mary,\ ACME),$ $collaborates\_with(Mary, John),$ $collaborates\_with(Alice,\ Mary),$ $collaborates\_with(Steve,\ John)\}$.

Any exploration task of the form $ET^{\bar{V},\bar{I}} = \langle \bar{V},\ \bar{I},\ ds\rangle$ represents the task of retrieving from the dataset ds all the data about any instance, no matter the language in which such data are expressed. For example, if $ET_1^{\bar{V},\bar{I}} = \langle \bar{V},\ \bar{I},\ ds\rangle$ (ds specified above), we have $SOL(ET_1^{\bar{V},\bar{I}}) = ds$.[1]

3.2 RDF-OWL Exploration Tasks: Exploration Tasks for RDF Datasets that Use OWL Ontologies

The formal definitions stated in Section 3.1 can be instantiated to a specific kind of exploration tasks, which is relevant for both Semantic Web and Linked Data: the case in which the dataset is an RDF dataset, possibly using terms from OWL ontologies for expressing (part of) its data.

RDF – Resource Description Framework – (W3C, 2014) is a model for data representation based on *triples*. A triple $\langle x\ p\ y\rangle$ represents a data element where x is an RDF resource (i.e., anything that can be talked about), p is a property and y can be either a resource or a value. The meaning of a triple is that the resource x is associated with y through the property p (e.g., $\langle Mary\ works\_at\ ACME\rangle$ means that the resource *Mary* is associated with the resource *ACME* through the property *works_at*; $\langle Mary\ age\ 45\rangle$ means that the resource *Mary* is associated with the value 45 through the property *age*).

OWL – Web Ontology Language – (W3C, 2012) is a formal language for representing ontologies in a standard and machine-readable way. OWL allows one to describe *instances* (e.g., *Mary*, *ACME*, etc.) by grouping them in classes (e.g., *Woman*, *Company*, etc.), by linking them through binary *object properties* (e.g., *works_at*, *collaborates_with*, etc.), by associating them with values through binary *data properties* (e.g., *age*, etc.). An OWL ontology (basically, a set of axioms) can be seen as a formal description of the semantics of a vocabulary whose terms refer to instances, classes, object properties, and data properties.

In the RDF-OWL setting, the definition of an exploration task is instantiated as follows:

Definition 3.3. *An RDF-OWL Exploration Task is an exploration task (Definition 3.1)* RO-ET $= \langle V = C \cup OP \cup DP \cup \{$rdf:type, owl:sameAs$\},\ I,\ ds\rangle$*, where:*

1. *C is a possibly empty and possibly infinite set of OWL class names;*
2. *OP is a possibly empty and possibly infinite set of OWL object properties names;*
3. *DP is a possibly empty and possibly infinite set of OWL data properties names;*
4. *I is a non-empty and possibly infinite set of RDF resources, representing instances;*
5. *ds is an RDF dataset.*

The definition above explicitly distinguishes the different categories of (logical) terms by means of which instances can be described in OWL (i.e., class, object property and data property names). Moreover, the vocabulary always includes the pre-defined properties rdf:type and owl:sameAs, expressing the membership of an instance to a class and the equivalence between two instances, respectively.

The definition of the solution to an exploration task (Definition 3.2) still holds for RDF-OWL Exploration Tasks. However, in many cases, solving an RDF-OWL Exploration Task may be difficult or even unfeasible, in practice. The main source of difficulty being that, in general, OWL data may be expressed by means of complex class expressions, negative properties, and other OWL constructs, which make it difficult (if not impossible) to compute solutions to RDF-OWL Exploration Tasks, especially if the dataset ds is huge and only remotely accessible through SPARQL queries.

For practical purposes, we thus define the notion of Restricted Solution to an RDF-OWL Exploration Task, as follows:

Definition 3.4. *A Restricted Solution to an RDF-OWL Exploration Task* RO-ET $= \langle V = C \cup OP \cup DP \cup \{$rdf:type, owl:sameAs$\},\ I,\ ds\rangle$ *is the set* R-SOL(RO-ET) = R-SOL$_C$(RO-ET) $\cup$ R-SOL$_{OP}$(RO-ET) $\cup$ R-SOL$_{DP}$(RO-ET) $\cup$ R-SOL$_{sameAs}$(RO-ET)*, where:*

1. R-SOL$_C$(RO-ET) $= \{\langle i$ rdf:type $c\rangle \in ds / i \in I \wedge c \in C\}$*;*
2. R-SOL$_{OP}$(RO-ET) $= \{\langle i\ op\ j\rangle \in ds / op \in OP \wedge (i \in I \vee j \in I)\}$*;*
3. R-SOL$_{DP}$(RO-ET) $= \{\langle i\ dp\ v\rangle \in ds / i \in I \wedge dp \in DP\}$*;*

[1] It is worth noting that, in general, for an exploration task $ET^{\bar{V},\bar{I}} = \langle \bar{V}, \bar{I}, ds\rangle$ it can be $SOL(ET_1^{\bar{V},\bar{I}}) \neq ds$, since all the data not describing instances possibly present in ds (e.g. the metadata about the terms of the language employed in ds) should be filtered out in the solution.

4. R-SOL$_{sameAs}$(RO-ET) $= \{\langle i$ owl:sameAs $j\rangle \in ds/i \in I \vee j \in I\}$;

The definition above takes into consideration only those RDF triples that express classes (the R-SOL$_C$(RO-ET) set), positive object properties (R-SOL$_{OP}$(RO-ET)), positive data properties (R-SOL$_{DP}$(RO-ET)) and identity (R-SOL$_{sameAs}$(RO-ET)) assertions on instances. Moreover, the considered class assertions have the form $\langle i$ rdf:type $c\rangle$, where c is a class name. Therefore, the other kinds of OWL assertions on instances that may be present in an RDF dataset (i.e., negative properties, difference among individuals, and complex class expression assertions on instances), are ruled out.

The above-stated definitions immediately entail the two following properties:

Proposition 3.2. *Any RDF-OWL Exploration Task always has one and only one (possibly empty) Restricted Solution.*

Proposition 3.3. *If* RO-ET $= \langle V = C \cup OP \cup DP, I, ds\rangle$ *is an RDF-OWL Exploration Task, it holds that* R-SOL(RO-ET) $\subseteq$ *SOL*(RO-ET).

Intuitively, Proposition 3.3 states that every restricted solution to an RDF-OWL exploration task is also a solution to it.

As an example, let us consider the dataset ds in Section 3.1. If we interpret the unary predicates as class names, the predicate *age* as a data property name, and the other predicates as object property names, we can easily rewrite ds into an RDF dataset ds_{RDF}, where, for example, "*Woman*(*Mary*)" and "*age*(*Mary*,45)" become "$\langle Mary$ rdf:type $Woman\rangle$" and "$\langle Mary\ age\ 45\rangle$" respectively). If $C_1 = \{Woman, Man, Factory\}$, $OP_1 = \{works\_at, collaborates\_with\}$, $DP_1 = \{age\}$, $V_1 = C_1 \cup OP_1 \cup DP_1 \cup \{$rdf:type, owl:sameAs$\}$ and $I_1 = \{Mary, Alice, Bob\}$, we can specify the RDF-OWL Exploration Task RO-ET$_1 = \langle V_1, I_1, ds_{RDF}\rangle$. The restricted solution R-SOL(RO-ET$_1$) to this exploration task is the union of the following sets:

1. R-SOL$_C$(RO-ET$_1$) $= \{\langle Mary$ rdf:type $Woman\rangle$, $\langle Alice$ rdf:type $Woman\rangle\}$;
2. R-SOL$_{OP}$(RO-ET$_1$) $= \{\langle Mary\ works\_at\ ACME\rangle$, $\langle Alice\ collaborates\_with\ Mary\rangle$, $\langle John\ collaborates\_with\ Mary\rangle\}$;
3. R-SOL$_{DP}$(RO-ET$_1$) $= \{\langle Mary\ age\ 45\rangle\}$;
4. R-SOL$_{sameAs}$(RO-ET$_1$) $= \emptyset$.

In this specific case, we have R-SOL(RO-ET$_1$) $=$ *SOL*(RO-ET$_1$) (i.e. the restricted solution is also the solution).

3.3 Computing Restricted Solutions to RDF-OWL Exploration Tasks

For practical purposes, in the following, we consider only RDF-OWL Exploration Tasks RO-ET $= \langle V = C \cup OP \cup DP \cup \{$rdf:type, owl:sameAs$\}, I, ds\rangle$ in which the sets C, OP, DP and I are either finite or they are, respectively, the set $\overline{C}$ of all possible class names, $\overline{OP}$ of all possible object property names, $\overline{DP}$ of all possible data property names and $\overline{I}$ of all possible instances.

The Restricted Solution to RO-ET can be computed by means of SPARQL queries.

SPARQL – SPARQL Protocol and RDF Query Language – (W3C, 2013) offers a language to query RDF datasets. The core part of a SPARQL query is represented by a set of triple patterns that express the main conditions that the pieces of data must meet. Each query execution matches these patterns with the data in order to single out all the possible pattern instantiations. For instance, if we consider the above-specified dataset ds, the triple patterns $[\langle ?x\ collaborates\_with\ ?y\rangle, \langle ?y\ works\_at\ ACME\rangle]$[2] can be instantiated in the two following ways: $[\langle John\ collaborates\_with\ Mary\rangle, \langle Mary\ works\_at\ ACME]$ and $[\langle Alice\ collaborates\_with\ Mary\rangle, \langle Mary\ works\_at\ ACME\rangle]$. Pattern instantiations can be restricted by means of suitable filters. For instance, the SPARQL filter "*FILTER EXISTS*$\{?x$ rdf:type $Woman\}$" would retain only the latter instantiation. SPARQL offers four query forms, including *ASK* and *SELECT*. The former simply tests whether the queried dataset provides at least one instantiation for the specified set of triple patterns (satisfying the optional filters); the latter returns as input a (combination of a) subset of the variable bindings in the pattern instantiations (that satisfy the optional filters).

In order to specify how this computation can be done, we consider the restricted solutions' subsets listed in Definition 3.4 and show how to calculate them; with $RES(Q)$ we indicate the result of a SPARQL query Q.

The set R-SOL$_C$(RO-ET) of *class assertions* is computed as follows:

R-SOL$_C$(RO-ET) $=$

- $\emptyset$, if $C = \emptyset$;
- $\{\langle i$ rdf:type $c\rangle / i \in I \wedge c \in C \wedge RES(ASK\ WHERE\{i\ a\ c.\}) == true\}$,

[2]Elements preceded by a question mark "?" denote variables.

$\underline{\text{if } C \neq \emptyset \wedge C \neq \overline{C} \wedge I \neq \overline{I}}$;

- $\{\langle ?x \text{ rdf:type } c \rangle / c \in C \wedge ?x \in RES(SELECT\ ?x\ WHERE\ \{?x\ a\ c.\})$, $\underline{\text{if } C \neq \emptyset \wedge C \neq \overline{C} \wedge I = \overline{I}}$;
- $\{\langle i \text{ rdf:type } ?y \rangle / i \in I \wedge ?y \in RES(SELECT\ ?y\ WHERE\ \{i\ a\ ?y.\})$, $\underline{\text{if } C = \overline{C} \wedge I \neq \overline{I}}$;
- $\{\langle ?x \text{ rdf:type } ?y \rangle / \langle ?x\ ?y \rangle \in RES(SELECT\ ?x\ ?y$
  $WHERE\ \{?x\ a\ ?y.$
  $FILTER\ (EXISTS\ \{?x\ a \text{ owl:NamedIndividual}\}\ ||$
  $EXISTS\ \{?y\ a \text{ owl:Class}\})\}$,
  $\underline{\text{if } C = \overline{C} \wedge I = \overline{I}}$.

Starting from the first case above, if no class name is specified (i.e., $C = \emptyset$), no class assertion is retrieved from the dataset. If non-empty and finite sets of class names and instances are specified ($C \neq \emptyset \wedge C \neq \overline{C} \wedge I \neq \overline{I}$) an ASK SPARQL query for each pair in $I \times C$ is performed to check whether the corresponding class assertion is present in the dataset. If a non-empty and finite set of class names is specified for any possible instance ($C \neq \emptyset \wedge C \neq \overline{C} \wedge I = \overline{I}$), all the class assertions involving a specified class name is retrieved, no matter the instance it refers to. If the set of all possible class names is specified for a finite set of instances ($C = \overline{C} \wedge I \neq \overline{I}$), all the class assertions on the specified instances are retrieved, no matter the employed class name. If the sets of all possible class names and instances are specified ($C = \overline{C} \wedge I = \overline{I}$), all the class assertions on instances present in the dataset are retrieved. In this last case, it should be noted that all the class assertions possibly present in a RDF-OWL dataset that do not refer to instances (i.e. to OWL named individuals) should be ruled out. This is the case, for example, of those assertions that specify that an ontology element is an *owl* : *Class*, an *owl* : *DatatypeProperty*, etc. Here we assume that the dataset explicitly specifies the OWL type at least for the individuals it describes or for the classes it employs. If this is the case, we can specify a simple filter that rules out unwanted triples (otherwise, we would need more elaborated filters).

The set R-SOL$_{OP}$(RO-ET) of *object property assertions* is computed as follows:

R-SOL$_{OP}$(RO-ET) =

- $\emptyset$, $\underline{\text{if } OP = \emptyset}$;
- $\{\langle i\ op\ ?y \rangle / i \in I \wedge op \in OP \wedge\ ?y \in RES(SELECT\ ?y\ WHERE\ \{i\ op\ ?y.\})\}$
  $\cup$
  $\{\langle ?x\ op\ i \rangle / i \in I \wedge op \in OP \wedge\ ?x \in RES(SELECT\ ?x\ WHERE\ \{?x\ op\ i.\})\}$, $\underline{\text{if } OP \neq \emptyset \wedge OP \neq \overline{OP} \wedge I \neq \overline{I}}$;
- $\{\langle ?x\ op\ ?y \rangle / op \in OP \wedge \langle ?x\ ?y \rangle \in RES(SELECT\ ?x\ ?y\ WHERE\ \{?x\ op\ ?y.\})\}$, $\underline{\text{if } OP \neq \emptyset \wedge OP \neq \overline{OP} \wedge I = \overline{I}}$;
- $\{\langle i\ ?p\ ?y \rangle / i \in I \wedge \langle ?p\ ?y \rangle \in RES(SELECT\ ?p\ ?y\ WHERE\ \{i\ ?p\ ?y.\})\}$
  $\cup$
  $\{\langle ?x\ ?p\ i \rangle / i \in I \wedge \langle ?x\ ?p \rangle \in RES(SELECT\ ?x\ ?p\ WHERE\ \{?x\ ?p\ i.\})\}$, $\underline{\text{if } OP = \overline{OP} \wedge I \neq \overline{I}}$;
- $\{\langle ?x\ ?p\ ?y \rangle / \langle ?x\ ?p\ ?y \rangle \in RES(SELECT\ ?x\ ?p\ ?y\ WHERE\ \{?x\ ?p\ ?y.$
  $FILTER\ (EXISTS\ \{?x\ a \text{ owl:NamedIndividual}\}||$
  $EXISTS\ \{?p\ a \text{ owl:ObjectProperty}\}||$
  $EXISTS\ \{?y\ a \text{ owl:NamedIndividual}\})\}$,
  $\underline{\text{if } OP = \overline{OP} \wedge I = \overline{I}}$;

If no object property name is specified (i.e. $OP = \emptyset$), no object property assertion is retrieved from the dataset. If non-empty and finite sets of object property names and instances are specified ($OP \neq \emptyset \wedge OP \neq \overline{OP} \wedge I \neq \overline{I}$), then each triple that represents an object property assertion through a specified object property and that is about a specified instance (which can be either subject or object of the triple) is retrieved. If a non-empty and finite set of object property names is specified for any possible instance ($OP \neq \emptyset \wedge OP \neq \overline{OP} \wedge I = \overline{I}$), then all the object property assertions involving a specified object property name is retrieved, no matter the instances it refers to. If the set of al possible object property names is specified for a finite set of instances ($OP = \overline{OP} \wedge I \neq \overline{I}$), all the object property assertions on the specified instances are retrieved, no matter the employed object property name and the subject or object role of the specified instances in the triples. If the sets of all possible object property names and instances are specified ($OP = \overline{OP} \wedge I = \overline{I}$), all the object property assertions on instances present in the dataset are retrieved. As in the case of class assertions, also in this case suitable filters are specified to rule out unwanted triples.

Sets R-SOL$_{DP}$(RO-ET) – data property assertions – and R-SOL$_{sameAs}$(RO-ET) – identity assertions on instances – mentioned in Definition 3.4, are computed in a similar way.

3.4 The Role of LDscout in PRiSMHA

Figure 1 shows an overview of PRiSMHA architecture, focusing on the role that LDscout plays in it. We only provide here a very quick sketch of this architecture, in order to describe the way we are using LDscout in the project: a detailed description of the other mentioned modules is out of the scope of this paper and can be found in (Caserio et al., 2017).

The system offers a crowd-sourcing platform enabling users to specify ontology-based formal semantic representations of the content of historical documents. Following a quite standard approach – e.g., (Cybulska and Vossen, 2011), (Sprugnoli and Tonelli, 2016) – such semantic representations are centered on the notion of *event*. Basically, events are described by means of their types, the places where they occur, the time when they happen and the entities that participate in them. These representations are based on the Historical Event Representation Ontology (HERO) (Caserio et al., 2017) and are stored in an *RDF triplestore*.

The crowd-sourcing platform relies on the support provided by the *Information Extraction module* and the *LOD mining module*. The Information Extraction module analyzes texts (when available) extracting information about entities (people, organizations, places), which are used to support users in the identification of relevant events and participants within the documents (Rovera et al., 2017).

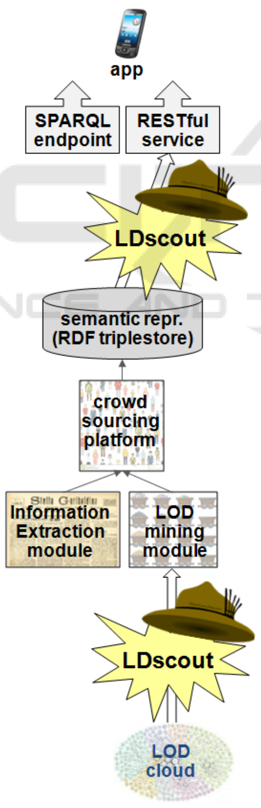

Figure 1: The role of LDscout in PRiSMHA architecture.

The LOD mining module queries LOD datasets in order to get more information about the entities identified by users in the documents. For example, if the user is reading a text about CGIL (the Italian General Confederation of Labour), and needs some more information about it, the LOD mining module can get such information from DBpedia (dbpedia.org) and provide it to the user. The retrieved information (e.g., that the acronym means "Confederazione Generale Italiana del Lavoro", that it is an Italian Trade Union, and it was founded in June 1944, etc.) is represented as a set of RDF triples, expressed in terms of the HERO vocabulary, so that it can be immediately used to enrich the semantic representation under construction on the crowd-sourcing platform.

In order to query LOD datasets and get results immediately usable on the PRiSMHA platform, the LOD mining module exploits LDscout APIs. For configuring the suitable Exploration Task in such a way that ensures results (Exploration Task Solutions) in terms of HERO, the LOD mining module relies on a set of mappings between: (a) HERO classes/properties and the vocabulary used in the selected dataset; (b) instances in the PRiSMHA triplestore and instances in the selected dataset. In the current proof-of-concept prototype, such mappings ensure interoperability with DBpedia and they have been manually encoded. However, if ontological mappings (i.e., those concerning classes and properties) can be seen as part of the design and implementation of the ontology interoperability, for instance mappings automatic tools such as LIMES (see Section 2) should be taken into account, in order to guarantee the scalability of the approach.

The RDF dataset produced by the crowd-sourcing platform is accessible through a standard *SPARQL endpoint*. However, in order to hide SPARQL complexity, providing a much easier to use interface, the dataset can also be accessed through a *RESTful service*, implemented thanks to LDscout APIs. In this case, the Exploration Task is defined on the dataset containing the semantic representation produced by the platform, enabling software clients to easily configure their queries with different sets of HERO classes and properties, and different sets of instances.

The availability of a RESTful service to access the semantic representations greatly simplifies the exploitation of the PRiSMHA dataset by third-party applications that aims at exploiting our data for different purposes, ranging from history-aware tourist guides to education-oriented tools.

4 CONCLUSIONS

In this paper, we presented a general model handling exploration tasks over datasets and we described its instantiation in LDscout, a Java library for querying RDF triplestores by specifying the vocabulary (ontology) and the set of instances the client is interested in.

To assess the usefulness of LDscout, we also showed how we used it within the PRiSMHA platform. With respect to this usage, the first improvement we are working on concerns the mappings between instances in our triplestore and instances in LOD sets: we are investigating the exploitation of automatic instance mapping tools, such as LIMES (Ngonga-Ngomo and Auer, 2011) in order to overcome the work overload of manually encoding such mappings.

Moreover, an interesting enhancement is represented by endowing LDscout with a friendly web-based user interface, available for human users, and not only for software clients.

ACKNOWLEDGEMENTS

This work has been partially supported by Compagnia di San Paolo and Università di Torino within the PRiSMHA project.

REFERENCES

Alexiev, V. (2016). Linked Open Data for cultural heritage and digital humanities. *Ontotext Blog*, September 28.

Berners-Lee, T., Hollenbach, J., Lu, K., Presbrey, J., Prud-hommeaux, E., and Schraefel, M. M. C. (2008). Tabulator redux: Browsing and writing linked data. In *Proc. LDOW 2008*.

Caserio, M., Goy, A., and Magro, D. (2017). Smart access to historical archives based on rich semantic metadata. In *Proc. KMIS 2017*, pages 93–100. SciTePress.

Cybulska, A. and Vossen, P. (2011). Historical event extraction from text. In *Proc. LaTeCH 2011*, pages 39–43.

Edelstein, J., Galla, L., Li-Madeo, C., Marden, J., Rhonemus, A., and Whysel, N. (2013). Linked Open Data for cultural heritage: Evolution of an information technology. In *Proc. 31st ACM International Conference on Design of Communication*. Association for Machine Computing.

Fafalios, P. and Tzitzikas, Y. (2013). X-ENS: Semantic enrichment of web search results at real-time. In *Proc. SIGIR 2013*, pages 1089–1090.

Goy, A., Damiano, R., Loreto, F., Magro, D., Musso, S., Radicioni, D., Accornero, C., Colla, D., Lieto, A., Mensa, E., Rovera, M., Astrologo, D., Boniolo, B., and D'ambrosio, M. (2017). Prismha (providing rich semantic metadata for historical archives). In *Proc. Contextual Representation of Objects and Events in Language (CREOL 2017)*.

Goy, A., Magro, D., and Rovera, M. (2015). Ontologies and historical archives: A way to tell new stories. *Applied Ontology*, 10(3-4):331–338.

Graub, M. A. N. B. C., Kharlamovb, E., Marciuška, S., and Zheleznyakov, D. (2016). Faceted search over RDF-based knowledge graphs. *Web Semantics: Science, Services and Agents on the World Wide Web*, 37-38:55–74.

Haag, F., Lohmann, S., Siek, S., and Ertl, T. (2015). Visual querying of linked data with QueryVOWL. In *Joint Proc. SumPre 2015 and HSWI 2014-2015*. CEUR.

Hahn, R., Bizer, C., Sahnwaldt, C., Herta, C., Robinson, S., Bürgle, M., Düwiger, H., and Scheel, U. (2010). Faceted Wikipedia search. In *Proc. BIS 2010*, pages 1–11.

Heath, T. and Bizer, C. (2011). *Linked Data: Evolving the Web into a Global Data Space*. Morgan & Claypool.

Heim, P., Ertl, T., and Ziegler, J. (2010). Facet graphs: Complex semantic querying made easy. In *Proc. ESWC 2010*, pages 288–302.

Heim, P., Ziegler, J., and Lohmann, S. (2008). gfacet: A browser for the web of data. In *IMC-SSW 2008*, pages 49–58.

Hildebrand, M., van Ossenbruggen, J., and Hardman, L. (2006). /facet: A browser for heterogeneous semantic web repositories. In *Proc. ISWC 2006*, pages 272–285.

Huynh, D. F. and Karger, D. R. (2009). Parallax and companion: Set-based browsing for the data web. In *Proc. WWW 2009*. ACM.

Hyvönen, E., Mäkelä, E., Salminen, M., Valo, A., Viljanen, K., Saarela, S., Junnila, M., and Kettula, S. (2005). Museumfinland - finnish museums on the semantic web. *Journal of Web Semantics*, 3(2-3):224–241.

Kurki, J. and Hyvönen, E. (2010). Collaborative metadata editor integrated with ontology services and faceted portals. In *Proc. ORES 2010*.

Navarro, J. F. G., Villamar, V. A. L., Srinivasan, J., Perry, M., Das, S., and Wu, Z. (2015). Exploring large RDF datasets using a faceted search. In *Proc. ISWC 2015 - Posters and Demos*. CEUR.

Ngonga-Ngomo, A. C. (2012). On link discovery using a hybrid approach. *Journal on Data Semantics*, 1:203–217.

Ngonga-Ngomo, A. C. and Auer, S. (2011). LIMES - a time-efficient approach for large-scale link discovery on theweb of data. In *Proc. IJCAI 2011*, pages 2312–2317.

Ngonga-Ngomo, A. C., Auer, S., Lehmann, J., and Zaveri, A. (2014). Introduction to linked data and its lifecycle on the web. In Koubarakis, M., Stamou, G., Stoilos, G., Horrocks, I., Kolaitis, P., Lausen, G., and Weikum, G., editors, *Proc. Reasoning Web 2014, LNCS 8714*, pages 1–99. Springer.

Ouksili, H., Kedad, Z., Lopes, S., and Nugier, S. (2016). Pattern-based keyword search on RDF data. In *Proc. ESWC Satellite Events 2016*, pages 30–34.

Potoniec, J., Jakubowski, P., and Ławrynowicz, A. (2017). Swift linked data miner: Mining OWL 2 EL class expressions directly from online RDF datasets. *Web Semantics: Science, Services and Agents on the World Wide Web*, 46-47:31–50.

Rovera, M., Nanni, Ponzetto, S. P., and Goy, A. (2017). Domain-specific named entity disambiguation in historical memoirs. In *Proc. CLiC-it 2017*. CEUR.

Russell, A. and Smart, P. R. (2010). NITELIGHT: A graphical editor for SPARQL queries. In *Proc. ISWC 2008 - Posters and Demos*.

Soylu, A., Giese, M., Jimenez-Ruiz, E., Vega-Gorgojo, G., and Horrocks, I. (2016). Experiencing optiquevqs: a multi-paradigm and ontology-based visual query system for end users. *Universal Access in the Information Society*, 15(1):129–152.

Sprugnoli, R. and Tonelli, S. (2016). One, no one and one hundred thusand events: Defining and processing events in an inter-disciplinary perspective. *Natural Language Engineering*, 23(4):485–506.

Tunkelang, D. (2009). *Faceted Search, Synthesis Lectures on Information Concepts, Retrieval, and Services*. Morgan & Claypool.

Tzitzikas, Y., Manolis, N., and Papadakos, P. (2017). Faceted exploration of RDF/S datasets: A survey. *Journal of Intelligent Information Systems*, 48(2):329–364.

Vega-Gorgojo, G., Giese, M., and Slaughter, L. (2017). Exploring semantic datasets with RDF surveyor. In *Proc. ISWC 2017 - Posters and Demos and Industry Tracks*.

Vega-Gorgojo, G., Slaughter, L., Giese, M., Heggestøyl, S., Klüwer, J. W., and Waaler, A. (2016). Pepesearch: Easy to use and easy to install semantic data search. In *Proc. ESWC 2016 Satellite Events*, pages 146–150.

Volz, J., Bizer, C., Gaedke, M., and Kobilarov, G. (2009). Discovering and maintaining links on the web of data. In *Proc. ISWC 2009*, pages 650–665.

W3C (2012). OWL 2 Web Ontology Language primer (second edition).

W3C (2013). SPARQL 1.1 overview.

W3C (2014). RDF 1.1 primer.