Query-Oriented Summarization of RDF Graphs

Šejla Čebirić, François Goasdoué, Ioana Manolescu

Query-Oriented Summarization of RDF Graphs

Šejla Čebirić
INRIA & U. Paris-Sud, France
sejla.cebiric@inria.fr

François Goasdoué
U. Rennes 1 & INRIA, France
fg@irisa.fr

Ioana Manolescu
INRIA & U. Paris-Sud, France
ioana.manolescu@inria.fr

ABSTRACT

The Resource Description Framework (RDF) is the W3C's graph data model for Semantic Web applications. We study the problem of RDF graph summarization: given an input RDF graph $\mathtt{G}$, find an RDF graph $\mathtt{S_G}$ which summarizes $\mathtt{G}$ as accurately as possible, while being possibly orders of magnitude smaller than the original graph. Our summaries are aimed as a help for query formulation and optimization; in particular, querying a summary of a graph should reflect whether the query has some answers against this graph. We introduce two summaries: a *baseline* which is compact and simple and satisfies certain accuracy and representativeness properties, but may oversimplify the RDF graph, and a *refined* one which trades some of these properties for more accuracy in representing the structure. The demonstration will allow the audience to compute such summaries out of a large variety of datasets, and explore their usage for data exploration and query optimization. The demonstration will also be presented in [4]; its main ideas appear in [5].

1. INTRODUCTION

The Resource Description Framework (RDF) is a graph-based data model promoted by the W3C as the standard for Semantic Web applications. Its associated query language is SPARQL.

RDF graphs are often *large* and *varied*, produced by applications ranging from scientific data, to social or online media, government data etc.; the Linked Data Catalog project[1] provides many interesting examples. They are *heterogeneous*, i.e., resources described in an RDF graph may have very different sets of properties. An RDF graph may assign *types* to resources, e.g., *Alice is of type Student*, however an object may have no type, or on the contrary have several types (which may or may not be related to each other).

RDF Schema (RDFS) may optionally be attached to an RDF graph, to enhance the description of its resources. RDFS statements may specify for instance that any resource of type *Student* is also of type *Person*, or that any *Friend* of somebody is a *LivingBeing*. Such statements also entail that in an RDF graph, some data is **implicit**: for instance, based on those above, one may derive that *Alice* is of type *Person*. According to the W3C RDF and SPARQL specification, **the semantics of an RDF graph comprises both its explicit and implicit data**; in particular, SPARQL query answers must be computed *reflecting both the explicit and implicit data*, even if the latter is not physically stored.

The RDF features mentioned above make it an extremely rich and flexible format for encoding data and/or application domain knowledge. They also make RDF graphs complex, both structurally and conceptually. It is intrinsically hard to get familiar with a new RDF dataset, especially if an RDF schema is not available for it, or if a schema is available but only characterizes a small part of the data, i.e., only few of the graphs' resources have types. Unfortunately, these situations occur quite frequently in practice, e.g., [20].

In this work, we study the problem of *RDF summarization*, that is: given an input RDF graph $\mathtt{G}$, find an RDF graph $\mathtt{S_G}$ which *summarizes* $\mathtt{G}$ *as accurately as possible, while being possibly orders of magnitude smaller* than the original graph. Such a summary can be used in a variety of contexts: to help an RDF application designer get acquainted with a new dataset, as a first-level user interface, or as a support for query optimization as typically used in semi-structured graph data management [10] etc. Our approach is *query-oriented*, i.e., a summary should enable static analysis and help formulating and optimizing queries; for instance, querying a summary of a graph should reflect whether the query has some answers against this graph, or finding a simpler way to formulate the query etc. While semi-structured data summarization has been studied before (see Section 5), our work is the first focused on *partially explicit, partially implicit* RDF graphs.

In the sequel, Section 2 recalls the basics of the RDF data, schema and queries, and sets the requirements for our query-oriented RDF summaries. Section 3 describes two flavors of summaries: a *baseline* which is compact, simple and meets our requirements, at the expense of a potentially too high simplification of the graph, and a *refined* one which trades some of our requirements for more accuracy in representing the structure. Section 4 presents our demonstration scenario. We then discuss related work and conclude.

2. PRELIMINARIES

We introduce RDF graph and queries in Section 2.1, and

[1] `http://linkeddatacatalog.dws.informatik.uni-mannheim.de/`

BDA Septembre 2015, Île de Porquerolles, France.

Assertion	Triple	Relational notation
Class	s rdf:type o	o(s)
Property	s p o	p(s, o)

Constraint	Triple	OWA interpretation
Subclass	s rdfs:subClassOf o	$s \subseteq o$
Subproperty	s rdfs:subPropertyOf o	$s \subseteq o$
Domain typing	s rdfs:domain o	$\Pi_{domain}(s) \subseteq o$
Range typing	s rdfs:range o	$\Pi_{range}(s) \subseteq o$

Figure 1: RDF (top) & RDFS (bottom) statements.

requirements for our RDF summaries in Section 2.2.

2.1 RDF Graphs and Queries

An *RDF graph* (or *graph*, in short) is a set of *triples* of the form s p o, stating that the *subject* s has the *property* p, and the value of that property is the *object* o.

We consider only well-formed triples, as per the W3C's RDF specification, using uniform resource identifiers (URIs), typed or un-typed literals (constants), and *blank nodes* (unknown URIs or literals) corresponding to a form of incomplete information; they can be seen as some *unknown URI or literal tokens*.

Notations. We use s, p, and o in triples as placeholders. Literals are shown as strings between quotes, e.g., "*string*".

Figure 1 (top) shows how to use triples to describe resources, that is, to describe the type of a resource (unary relation) and a resource property (binary relation). The RDF standard provides a set of built-in classes and properties, as part of the rdf: and rdfs: pre-defined namespaces. We use these namespaces exactly for these classes and properties, e.g., rdf:type specifies the class(es) to which a resource belongs. *For brevity, we will sometimes use τ to denote* rdf:type.

For example, the RDF graph G shown below describes a book, identified by doi_1: its author (a blank node $\_{:}b_1$ related to the author name), title and date of publication.

$$\mathtt{G} = \begin{array}{l}\{\text{doi}_1 \text{ rdf:type } \texttt{Book},\ \text{doi}_1 \texttt{ writtenBy } \_{:}b_1,\\ \text{doi}_1 \texttt{ hasTitle } \text{“}\texttt{Port des Brumes}\text{”},\\ \_{:}b_1 \texttt{ hasName } \text{“}\texttt{G. Simenon}\text{”},\\ \text{doi}_1 \texttt{ publishedIn } \text{“}1932\text{”}\}\end{array}$$

RDF Schema allows enhancing the descriptions in RDF graphs by means of *RDFS triples*, declaring *semantic constraints* between the classes and the properties used in those graphs. Figure 1 (bottom) shows the allowed constraints and how to express them; *domain* and *range* denote respectively the first and second attribute of every property.

The RDFS constraints (Figure 1) are interpreted under the open-world assumption (OWA) [1]. For instance, given two relations R_1, R_2, the OWA interpretation of the constraint $R_1 \subseteq R_2$ is: *any tuple t in the relation R_1 is considered as being also in the relation R_2* or, in other words, the constraint *propagates t to R_2*. This leads to **implicit triples** which may be part of the RDF graph even though they are not explicitly present in it. An implicit triple can be obtained by an *immediate entailment step* based on (*i*) an RDFS constraint (of one of the four forms at the bottom of Figure 1), and (*ii*) either a second constraint (also called *schema triple*) or an RDF triple that is not a constraint (also termed *data triple*). A triple is *entailed by a graph* G, if and only if there is a sequence of applications of entailment rules

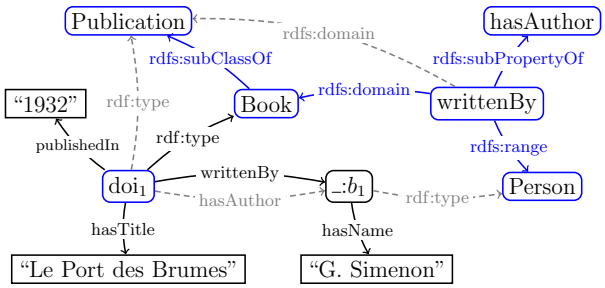

Figure 2: Sample RDF graph.

that leads from the graph to the triple (where at each step of the entailment sequence, the triples previously entailed are also taken into account). For instance, assume that the RDF graph G above is extended with the following constraints.

- books are publications:
  Book rdfs:subClassOf Publication
- writing something means being an author:
  writtenBy rdfs:subPropertyOf hasAuthor
- writtenBy is a relation between books and people:
  writtenBy rdfs:domain Book and
  writtenBy rdfs:range Person

The resulting graph is depicted in Figure 2. Its implicit triples are those represented by dashed-line edges.

Saturation. The immediate entailment rules allow defining the finite *saturation* (a.k.a. closure) of an RDF graph G, which is the RDF graph $\mathtt{G}^\infty$ defined as the fixed-point obtained by repeatedly applying entailment rules on G.

The saturation of an RDF graph is unique (up to blank node renaming), and does not contain implicit triples (they have all been made explicit by saturation). Clearly, a graph G entails a triple if and only if the triple belongs to $\mathtt{G}^\infty$. RDF entailment is part of the RDF standard; *the answers to a query posed on* G *must take into account all triples in* $\mathtt{G}^\infty$, since *the semantics of an RDF graph is its saturation.*

Queries. We consider the SPARQL dialect consisting of *basic graph pattern* (BGP) queries, a.k.a. conjunctive queries, widely considered in research but also in real-world applications [15]. A BGP is a set of *query triple patterns*, or query triples in short; each triple has a subject, property and object, some of which can be variables.

Notations. We use the notation $q(\bar{x}) \text{ :- } t_1, \ldots, t_\alpha$, where $\{t_1, \ldots, t_\alpha\}$ are query triple patterns; the query head variables $\bar{x}$ are called *distinguished variables*, and are a subset of the variables in $t_1, \ldots, t_\alpha$. For boolean queries $\bar{x}$ is empty. The head of q is $q(\bar{x})$, its body is $t_1, \ldots, t_\alpha$; x, y, z, etc. denote variables.

Query answering. The evaluation of a query q against G has access only to G's explicit triples, thus may lead to an incomplete answer; the complete answer is obtained by evaluating q against $\mathtt{G}^\infty$. For instance, the query below asks for name of the author of "Le Pont des Brumes":

$$q(x_3) \text{ :- } x_1 \texttt{ hasAuthor } x_2,\ x_2 \texttt{ hasName } x_3,\ x_1 \texttt{ hasTitle } \text{“}\texttt{Le Port des Brumes}\text{”}$$

Its answer against the graph in Figure 2 is $q(\mathtt{G}^\infty) = \{\langle \text{“G. Simenon”} \rangle\}$. Note that evaluating q only against G leads to the empty answer, which is obviously incomplete.

2.2 RDF Summary Requirements

We assume that *the summary* $\mathtt{S_G}$ *of an RDF graph* $\mathtt{G}$ *is an RDF graph itself*. Further, we require summaries to satisfy the following two conditions:

Completeness The saturation of the summary of $\mathtt{G}$ must be the same as the summary of its saturation $\mathtt{G}^\infty$, due to the semantics of an RDF graph being its saturation.

Schema independence It must be possible to summarize $\mathtt{G}$ whether or not it has associated RDFS triples.

The following properties are of a more quantitative nature:

Compactness The summary should be typically smaller than the RDF graph, ideally by orders of magnitude.

Representativeness The summary should not lose too much information from $\mathtt{G}$.

Accuracy The summary should avoid, to the extent possible, reflecting data that does not exist in $\mathtt{G}$.

A trade-off exists between compactness and representativeness, as the latter tends to require more information.

Criteria for representativeness and accuracy. Our query-oriented RDF graph summarization leads us to the following criteria. For *representativeness*, queries with results on $\mathtt{G}$ should also have results on the summary. Symmetrically, for *accuracy*, a query that can be matched on the summary, should also be matched on the RDF graph itself.

To formalize our criteria, we use $\mathcal{Q}$ to denote an *RDF query language (dialect)*; a concrete choice of such a dialect will shortly follow.

Definition 1. (Query-based representativeness) $\mathtt{S_G}$ *is* $\mathcal{Q}$*-representative of* $\mathtt{G}$ *if and only if for any query* $q \in \mathcal{Q}$ *such that* $q(\mathtt{G}^\infty) \neq \emptyset$*, we have* $q(\mathtt{S}_\mathtt{G}^\infty) \neq \emptyset$.

Note that *several graphs may have the same summary*, which corresponds to the intuition that a summary loses *some* of the information from the original graph. If two RDF graphs differ only with respect to such information, they have the same summary. We term *inverse set* of a summary $\mathtt{S_G}$, the set of all RDF graphs whose summary is $\mathtt{S_G}$. This leads to the accuracy criterion:

Definition 2. (Query-based accuracy) *Let* $\mathtt{S_G}$ *be a summary, and* $\mathcal{G}$ *the inverse set of* $\mathtt{S_G}$*. The summary* $\mathtt{S_G}$ *is* $\mathcal{Q}$-accurate *if for any query* $q \in \mathcal{Q}$ *such that* $q(\mathtt{S}_\mathtt{G}^\infty) \neq \emptyset$*, there exists* $\mathtt{G} \in \mathcal{G}$ *such that* $q(\mathtt{G}^\infty) \neq \emptyset$.

The above characterizes the accuracy of a summary with respect to *any graph it may correspond to*.

For the sake of compactness, we decide that the (voluminous) set of literals, along with subject and object URIs for non-τ triples from $\mathtt{G}$ should not appear in $\mathtt{S_G}$. However, given that property URIs are often specified in SPARQL queries [2], and that there are typically far less distinct property URIs than there are distinct subject or object URIs [20], property URIs should be preserved by the summary. This leads us to the following SPARQL dialect:

Definition 3. (Relational BGP) *A* relational BGP *(RBGP, in short) is a BGP query whose body has: (i) URIs in all the property positions, (ii) a URI in the object position of every* τ *triple, and (iii) variables in any other positions.*

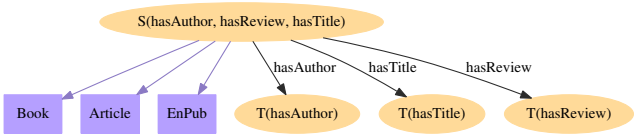

Figure 3: Baseline summary $\mathtt{B}_{\mathtt{G}_0}$ for the sample graph $\mathtt{G}_0$.

We define *RBGP representativeness* and *RBGP accuracy* by instantiating $\mathcal{Q}$ in Definition 1 and Definition 2, respectively, to RBGP queries (Definition 3).

3. RDF SUMMARIES

This section describes our RDF summary proposals.

3.1 Baseline Summary

We now introduce our first version of RDF graph summary, denoted *baseline* summary.

For this, we assume a function `newURI()` returning a fresh URI on each call. We call *data property* any property $\mathtt{p}$ occurring in the data component of $\mathtt{G}$, and different from τ. Further, for any data property $\mathtt{p}$, the *property source of* $\mathtt{p}$, denoted $\mathcal{S}(\mathtt{p})$, is a URI set by `newURI()`, and similarly, the *property target of* $\mathtt{p}$, denoted $\mathcal{T}(\mathtt{p})$, is a URI set by `newURI()`.

Definition 4. (Baseline summary) *Given an RDF graph* $\mathtt{G}$*, the baseline summary of* $\mathtt{G}$ *is an RDF graph* $\mathtt{B_G}$ *such that:*

Schema $\mathtt{B_G}$ *has the same schema triples as* $\mathtt{G}$.

DNT *(Data triples of* $\mathtt{B_G}$ *whose property is not* τ*) Let* $\mathtt{p}, \mathtt{p}_1, \mathtt{p}_2$ *be some data properties from* $\mathtt{G}$.

- **DNT1** *The triple* $\mathcal{S}(\mathtt{p})\ \mathtt{p}\ \mathcal{T}(\mathtt{p})$ *belongs to* $\mathtt{B_G}$;
- **DNT2** *if* $\mathtt{s}\ \mathtt{p}_1\ \mathtt{o}_1,\ \mathtt{s}\ \mathtt{p}_2\ \mathtt{o}_2 \in \mathtt{G}$*, then* $\mathcal{S}(\mathtt{p}_1) = \mathcal{S}(\mathtt{p}_2)$;
- **DNT3** *if* $\mathtt{s}_1\ \mathtt{p}_1\ \mathtt{o},\ \mathtt{s}_2\ \mathtt{p}_2\ \mathtt{o} \in \mathtt{G}$*, then* $\mathcal{T}(\mathtt{p}_1) = \mathcal{T}(\mathtt{p}_2)$;
- **DNT4** *if* $\mathtt{s}\ \mathtt{p}_1\ \mathtt{o}_1,\ \mathtt{o}_1\ \mathtt{p}_2\ \mathtt{o}_2 \in \mathtt{G}$*, then* $\mathcal{T}(\mathtt{p}_1) = \mathcal{S}(\mathtt{p}_2)$;

DT *(Data triples of* $\mathtt{B_G}$ *whose property is* τ*)*

- **DT1** *If* $\mathtt{s}\ \mathtt{p}\ \mathtt{o},\ \mathtt{s}\ \tau\ \mathtt{c}$ *are in* $\mathtt{G}$*, then* $\mathcal{S}(\mathtt{p})\ \tau\ \mathtt{c}$ *is in* $\mathtt{B_G}$;
- **DT2** *if* $\mathtt{s}\ \mathtt{p}\ \mathtt{o},\ \mathtt{o}\ \tau\ \mathtt{c}$ *are in* $\mathtt{G}$*, then* $\mathcal{T}(\mathtt{p})\ \tau\ \mathtt{c}$ *is in* $\mathtt{B_G}$;
- **DT3** *Let* n_τ *be set to* `newURI()`*. If* $\mathtt{s}\ \tau\ \mathtt{c} \in \mathtt{G}$*, and* $\nexists\, \mathtt{s}\ \mathtt{p}\ \mathtt{o} \in \mathtt{G}$ *and* $\nexists\, \mathtt{s}'\ \mathtt{p}\ \mathtt{s} \in \mathtt{G}$*, then* $n_\tau\ \tau\ \mathtt{c} \in \mathtt{B_G}$.

The baseline summary has the same schema as $\mathtt{G}$ (**Schema**), as well as a source and a target URI for each data property (**DNT1**). As soon as two data properties have the same subject and/or object, their corresponding source and target URIs are *the same* in $\mathtt{B_G}$ accordingly (**DNT2**-**DNT4**). If the subject (or the object) of a data property $\mathtt{p}$ is of type $\mathtt{c}$, then the $\mathtt{p}$ source (or target) URI is declared to be of type $\mathtt{c}$ in $\mathtt{B_G}$ (**DT1**-**DT2**). Finally, a single URI in $\mathtt{B_G}$ reflects all the subjects of τ triples in $\mathtt{G}$ that do not appear as subject or object of a data property in $\mathtt{G}$ (**DT3**).

For instance, consider the graph $\mathtt{G}_0$ describing the resources `doi1` to `doi3`:

doi1 τ Book	doi2 τ EnPub
doi1 hasTitle "T1"	doi2 τ Article
doi1 hasAuthor "A1"	doi2 hasTitle "T2"
doi1 hasAuthor "A2"	doi2 hasAuthor "A3"
doi3 hasTitle "T3"	doi2 hasReview "R1"

doi3 hasAuthor "A4"

where `EnPub` is the class of publications in English; we omitted a schema, to focus on the treatment of the data triples. Figure 3 depicts the baseline summary of $\mathtt{G}_0$; the rectangular

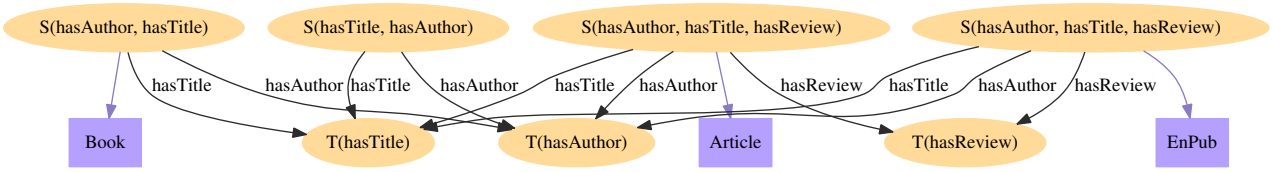

Figure 4: Refined summary $\mathtt{R}_{\mathtt{G}_0}$ for the sample graph $\mathtt{G}_0$.

nodes correspond to class URIs copied from $\mathtt{G}_0$, whereas the oval nodes are URIs created by `newURI()`.

Importantly, the baseline summary meets our requirements, as follows (the proofs can be found in [21]).

We say two summary graphs are *equivalent*, denoted $\equiv$, iff they are identical up to a bijection between their sets of URIs resulting from calls to `newURI()`. We meet the completeness requirement with the next proposition, stating the commutativity of saturation and summarization:

PROPOSITION 1. *Let $\mathtt{B}_\mathtt{G}$ be the baseline summary of $\mathtt{G}$, and $\mathtt{B}_{\mathtt{G}^\infty}$ the baseline summary of $\mathtt{G}^\infty$. Then: $(\mathtt{B}_\mathtt{G})^\infty \equiv \mathtt{B}_{\mathtt{G}^\infty}$.*

Regarding representativeness and accuracy, we show [21]:

PROPOSITION 2. *The baseline summary is (i) RBGP representative and (ii) RBGP accurate.*

It is easy to see that the baseline summary can be built in $O(|\mathtt{G}|^2)$ time. Its size is bounded by the size of $\mathtt{G}$'s schema to which we add (i) the number of data properties from $\mathtt{G}$ and (ii) the number of class assertions from $\mathtt{G}$.

3.2 Refined Summary

The baseline summary may unify property source and target URIs quite aggressively. For instance, if a *store* and a *person* both have a *zipcode*, they will lead to the same baseline URI (through rule **DNT2**), even though they are very different things.

To mitigate this issue, we designed a second flavor of summary of an RDF graph $\mathtt{G}$, termed *refined* and denoted $\mathtt{R}_\mathtt{G}$. For space reasons, the definition is delegated to [21]. Intuitively, the difference between the baseline and the refined summary is that the latter fuses data property source and/or target URIs *only if one resource in $\mathtt{G}$ that leads to their unification has no type at all*. For illustration, the refined summary $\mathtt{R}_{\mathtt{G}_0}$ of the same sample graph appears in Figure 4. The target URIs $\mathcal{T}(\mathtt{hasTitle})$, $\mathcal{T}(\mathtt{hasAuthor})$ and $\mathcal{T}(\mathtt{hasReview})$ are the same as in $\mathtt{B}_{\mathtt{G}_0}$; however, in $\mathtt{R}_{\mathtt{G}_0}$, four URIs have been created in the upper row to represent resources having both a title and an author, respectively (from left to right): resources of type `Book`, those having no type in $\mathtt{G}_0$, those of type `Article`, and those of type `EnPub`.

The refined summary commutes with saturation (given a graph $\mathtt{G}$, $(\mathtt{R}_\mathtt{G})^\infty = \mathtt{R}_{\mathtt{G}^\infty}$); it is also RGBP accurate [21]. It is *more accurate than the baseline*, as illustrated in Figure 4: the rightmost URI in $\mathtt{R}_{\mathtt{G}_0}$ shows that only resources of type `Article` or `EnPub` have reviews, whereas the baseline $\mathtt{B}_{\mathtt{G}_0}$ may lead one to believe that a resource of type `Book` also has a review. (Recall that this may happen in *some* graph $\mathtt{G}_1$ whose summary is $\mathtt{B}_{\mathtt{G}_0}$; it just does not happen in $\mathtt{G}_0$).

This extra accuracy comes at a cost. Computing the refined summary has $O(|\mathtt{G}|^5)$ complexity, which requires an efficient underlying system e.g., based on triple partitioning and indexing etc. The refined summary is representative for *all RBGPs which do not have more than one τ triple with the same subject*. This follows from a *graph homomorphism* from $\mathtt{G}^\infty$ to $(\mathtt{R}_\mathtt{G})^\infty$ [21].

An upper bound for its size is the number of classes in $\mathtt{G} \times$ the number of distinct data properties. This is significantly larger than for $\mathtt{B}_\mathtt{G}$, but in practice (i) the bound is seldom reached (ii) more accurate summaries are better appreciated by users getting acquainted with RDF graphs.

4. DEMONSTRATION SCENARIO

We demonstrate our Java tool (7.700 lines approx.) for computing baseline and refined RDF graph summaries. The tool issues queries that are executed by the underlying RDF store, in particular OpenLink Virtuoso Server (7.1), a PostgreSQL based RDF store complete with indexes etc., and a Hadoop-based RDF query processing platform [8].

Demo attendees will be able to: (i) pick an RDF graph $\mathtt{G}$ from a set including LUBM data, an RDF dump of DBLP, open data sets from the French INSEE (statistics) and IGN (geographic) institutes, as well as small hand-crafted examples chosen for their interest in illustrating summary features such as representativity, accuracy, dependence on the degree of saturation (recall that the saturated summary does not depend on whether $\mathtt{G}$ is saturated, but the non-saturated $\mathtt{B}_\mathtt{G}$ and $\mathtt{R}_\mathtt{G}$ do!) (ii) compute $\mathtt{B}_\mathtt{G}$ and $\mathtt{R}_\mathtt{G}$ using one of the systems; (iii) inspect the summary with the help of ATT's GraphViz/DOT-based GUI; (iv) trigger the saturation of the summary; (v) modify the graphs in the store and see the impact on the summaries; (vi) choose or write custom RBGP queries, comprising *property paths*, specified by regular expressions of SPARQL v1.1., and see how they *unfold* into unions of queries, or how the summary allows deciding that they have empty results. Step (vi) adapts Dataguide techniques [10] to our RDF-specific summaries.

5. RELATED WORK

OEM and XML summaries. Dataguides [10] were introduced to summarize semistructured OEM graphs, similar to RDF, but assumed to have a "root" node, from which all others are accessible; this may not hold for RDF. Dataguides construction has worst-case exponential time complexity, thus is not in general feasible. An algorithm is provided for building *strong* Dataguides, used as a basis for indexing. The 1-index [16] groups together OEM or XML nodes that are reachable by exactly the same set of paths. Later works focused on indexes for supporting XML path queries [6, 12], or path-based XML summarization into graphs [7]. All these works differ from ours, because the input is a tree or DAG and/or because it lacks types and implicit information.

Graph summarization. Graph summarization has been very intensively studied, in particular through mining or clustering; large-scale graph processing is also a hot topic. A large number of works build on the idea of Dataguides for graph data, oftentimes referred to as *structural indexes*, which bear a similarity to graph summaries, both being a re-

duced version of the input graph and collapsing nodes based on some common attributes.

Our focus is on *RDF graphs* with *implicit* data, for which we devised *query-oriented* summaries, which are RDF graphs themselves and may be computed on a variety of platforms.

Graph *cores* have been studied in [9]. A graph core `C` for a given graph `G` is a graph such that an isomorphism exists between `G` and `C`, and `C` is the smallest graph with this property. Neither our baseline summary $\mathtt{B_G}$ nor the refined summary $\mathtt{R_G}$ are cores of the incoming graph `G`, since we cannot guarantee a homomorphism from either summary to the graph `G`. In exchange, both summary versions we consider can be built in polynomial time in the size of `G`, while computing the core is much harder.

Bisimulation-based approaches group nodes by the similarity of their neighborhood [14, 18]. In [14], graphs from the LOD cloud are summarized, focusing on the distribution of classes and properties across LOD sources. The resulting *resource-oriented* summary comprises unlabeled edges. [18] constructs a structural index from property paths of some maximum length. The main problem with bisimulation is that as the size of the neighborhood increases, the size of bisimulation grows exponentially and can be as large as the input graph. Thus, as we aim for both complete and compact summaries, bisimulation is not a good fit.

To overcome the problem with bisimulation, [11] suggests locality-based summaries, generated by a graph partitioning algorithm. Nonetheless, a reduction of the input RDF graph is necessary which is achieved by removing triples having properties with literal values, thus resulting in an incomplete summary. Recall that we wish to represent queries comprising properties with literal values as well.

A *triple-oriented* structural index for RDF data is built in [17] as a non-RDF graph, where a node comprises a set of triples forming a data partition, while an edge describes the way in which triples from its adjacent nodes join. [19] proposes a tree RDF index, storing regions defined by center vertices, limited to property paths and built assuming that the input RDF graph is saturated.

In [13], RDF classes are inferred based on the common properties of resources. Thus, only common source patterns are analyzed, while the common targets and property paths are not considered. Further, the rdf:type properties that a dataset may comprise are simply ignored. [3] explores alternative RDF summaries w.r.t. graph homomorphism and trades precision for computing efficiency.

Finally, summarizing implicit data is not considered in any of these works.

6. CONCLUSION

The demonstration is focused on presenting our novel approach to RDF graph summarization in the form of baseline and refined summaries, produced for diverse RDF datasets and computed on a variety of platforms. We aim to illustrate various aspects of the summaries: summarization of triples that do not exist explicitly in the dataset and the natural tension between keeping the summaries compact, while retaining the full representativeness, resulting in the trade-offs in accuracy. Further, we showcase how the query orientation can be leveraged in query formulation and optimization, adapting the Dataguide techniques to the RDF realm.

Acknowledgments This work has been partially funded by the projects PIA Datalyse and DGA RAPID ODIN.

7. REFERENCES

[1] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison-Wesley, 1995.

[2] M. Arias, J. D. Fernández, M. A. Martínez-Prieto, and P. de la Fuente. An empirical study of real-world SPARQL queries. *CoRR*, abs/1103.5043, 2011.

[3] S. Campinas, R. Delbru, and G. Tummarello. Efficiency and precision trade-offs in graph summary algorithms. In *IDEAS*, 2013.

[4] S. Cebiric, F. Goasdoué, and I. Manolescu. Query-oriented summarization of RDF graphs. *PVLDB*, 8(12), 2015. To appear.

[5] S. Cebiric, F. Goasdoué, and I. Manolescu. Query-oriented summarization of RDF graphs. In *Data Science - 30th British International Conference on Databases, BICOD 2015, Edinburgh, UK, July 6-8, 2015, Proceedings*, pages 87–91, 2015.

[6] Q. Chen, A. Lim, and K. W. Ong. $D(K)$-index: An adaptive structural summary for graph-structured data. In *SIGMOD*, 2003.

[7] M. P. Consens, R. J. Miller, F. Rizzolo, and A. A. Vaisman. Exploring XML web collections with DescribeX. *ACM TWeb*, 4(3), 2010.

[8] F. Goasdoué, Z. Kaoudi, I. Manolescu, J.-A. Quiané-Ruiz, and S. Zampetakis. CliqueSquare: Flat Plans for Massively Parallel RDF Queries. In *ICDE*, 2015.

[9] C. Godsil and G. Royle. *Algebraic graph theory*. Springer-Verlag, 2001.

[10] R. Goldman and J. Widom. Dataguides: Enabling query formulation and optimization in semistructured databases. In *VLDB*, 1997.

[11] S. Gurajada, S. Seufert, I. Miliaraki, and M. Theobald. Using graph summarization for join-ahead pruning in a distributed RDF engine. In *Semantic Web Information Management Workshop (SWIM)*, pages 41:1–41:4, 2014.

[12] R. Kaushik, P. Bohannon, J. F. Naughton, and H. F. Korth. Covering indexes for branching path queries. In *SIGMOD*, 2002.

[13] K. Kellou-Menouer and Z. Kedad. A clustering based approach for type discovery in RDF data sources. In *15èmes Journées Francophones Extraction et Gestion des Connaissances, EGC 2015, 27-30 Janvier 2015, Luxembourg*, pages 471–472, 2015.

[14] S. Khatchadourian and M. P. Consens. ExpLOD: Summary-based exploration of interlinking and RDF usage in the linked open data cloud. In *ESWC*, 2010.

[15] D. Lanti, M. Rezk, G. Xiao, and D. Calvanese. The NPD benchmark: Reality check for OBDA systems. In *EDBT*, 2015.

[16] T. Milo and D. Suciu. Index structures for path expressions. In *ICDT*, 1999.

[17] F. Picalausa, Y. Luo, G. H. L. Fletcher, J. Hidders, and S. Vansummeren. A structural approach to indexing triples. In *ESWC*, pages 406–421, 2012.

[18] T. Tran, G. Ladwig, and S. Rudolph. Managing structured and semistructured RDF data using structure indexes. *IEEE Trans. Knowl. Data Eng.*, 25(9):2076–2089, 2013.

[19] O. Udrea, A. Pugliese, and V. S. Subrahmanian. GRIN: A graph based RDF index. In *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence, July 22-26, 2007, Vancouver, British Columbia, Canada*, pages 1465–1470. AAAI Press, 2007.

[20] Statistics on The Billion Triple Challenge Dataset. `gromgull.net/blog/2010/09/btc2010-basic-stats`, 2010.

[21] Technical report, 2015.