Query-Oriented Summarization of RDF Graphs

Šejla Čebirić, François Goasdoué, Ioana Manolescu

Query-Oriented Summarization of RDF Graphs

Šejla Čebirić[1], François Goasdoué[2,1], and Ioana Manolescu[1]

[1] INRIA and U. Paris-Sud, France {sejla.cebiric,ioana.manolescu}@inria.fr
[2] U. Rennes 1, Lannion, France fg@irisa.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 G, find an RDF graph S_G which summarizes G as accurately as possible, while being possibly orders of magnitude smaller than the original graph. Our approach is *query-oriented*, i.e., querying a summary of a graph should reflect whether the query has some answers against this graph. The summaries are aimed as a help for query formulation and optimization. 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.

1 Introduction

The Resource Description Framework (RDF) is a graph-based data model promoted by the W3C as the standard for Semantic Web applications; SPARQL is the W3C's standard language for querying RDF data.

RDF graphs are often *large* and *varied*, produced in a variety of contexts, e.g., scientific applications, social or online media, government data etc. They are *heterogeneous*, i.e., resources described in an RDF graph may have very different sets of properties. An RDF resource may have: no types, one or several types (which may or may not be related to each other). *RDF Schema* (RDFS) information may optionally be attached to an RDF graph, to enhance the description of its resources. Such statements also entail that in an RDF graph, some data is **implicit**. According to the W3C RDF and SPARQL specification, **the semantics of an RDF graph comprises both its explicit and its implicit data**; in particular, SPARQL query answers must be computed *reflecting both its explicit and implicit data*, even if the latter is not physically stored.

In this work, we study the problem of *RDF graph summarization*, that is: given an input RDF graph G, find an RDF graph S_G which *summarizes* 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 traditionally the case in semi-structured graph data management [3] etc. Our approach is *query-oriented*, i.e., querying a summary of a graph should reflect whether the query has some answers against this graph. The properties our summaries aim at are related to query processing, in particular enabling static analysis, query formulation

and optimization (i.e., deciding if a query is empty or finding a simpler way to formulate a query). While semi-structured data summarization has been studied before, our work is the first focused on *partially explicit, partially implicit* RDF graphs. Our ongoing technical report [5] provides proofs for the results presented here and a discussion of related work.

2 RDF Graphs and Summary Requirements

RDF graphs and queries An *RDF graph* (or *graph*) 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`. Triples are formed using uniform resource identifiers (URIs), typed or untyped literals (constants), and *blank nodes* (unknown URIs or literals) corresponding to incomplete information.We use `s`, `p`, and `o` in triples as placeholders. Literals are shown as strings between quotes, e.g., "*string*".

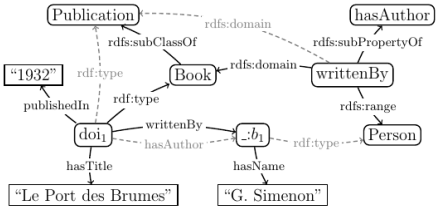

Fig. 1. Sample RDF graph

The RDF standard provides a set of built-in classes and properties in the rdf: and rdfs: pre-defined namespaces, e.g., triples of the form `s` rdf:type `o` specify the class(es) to which a resource belongs. *For brevity, we use* `type` *to denote* rdf:type. For example, the RDF graph `G` below describes a book, identified by $\texttt{doi}_1$: its author (a blank node $\_:b_1$ related to the author name), title and publication date.

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

RDF Schema triples allow enhancing the descriptions in RDF graphs by declaring *deductive constraints* between the graph classes and properties, namely: *subClassOf*, *subPropertyOf*, *domain* and *range*, where the latter two denote the first and second attribute of a property, respectively. Consequently, an RDF graph may have **implicit triples** even though they do not exist explicitly. For instance, assume 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 Fig. 1. Its implicit triples are those represented by dashed-line edges. Adding all the implicit triples to an RDF graph `G` leads to its *saturation* $\texttt{G}^\infty$, which is the RDF graph stating the semantics of `G`.

In this work, we consider *conjunctive* SPARQL queries, a.k.a. Basic Graph Pattern (BGP) queries. The evaluation of a query q against an RDF graph `G` based on `G`'s explicit triples may lead to an incomplete answer; the complete answer is obtained by evaluating q against $\texttt{G}^\infty$. E.g., consider:
$q(x_3)$:- x_1 `hasAuthor` x_2, x_2 `hasName` x_3, x_1 `hasTitle` “Le Port des Brumes”

Its answer against the graph in Fig. 1 is $q(\mathtt{G}^\infty) = \{\langle \text{“G. Simenon”} \rangle\}$. Note that evaluating q against $\mathtt{G}$ leads to an empty answer.

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 conditions: (*i*) The saturation of the summary of an RDF graph $\mathtt{G}$ must be the same as the summary of its saturation $\mathtt{G}^\infty$, since the semantics of an RDF graph is its saturation; (*ii*) The summary should be (if possible, much) smaller than the RDF graph; (*iii*) The summary should be *representative*: queries with results on $\mathtt{G}$ should also have results on the summary; (*iv*) The summary should be *accurate*: queries with results on the summary should reflect that such data existed indeed in the graph. To formalize these, let $\mathcal{Q}$ be a SPARQL dialect.

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*, since 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 $\mathtt{S_G}$, the set of all RDF graphs whose summary is $\mathtt{S_G}$. This leads to the accuracy criterion, with respect to *any graph a summary may correspond to*:

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* $q(\mathtt{G}^\infty) \neq \emptyset$*.*

For compactness, the (voluminous) set of literals, along with subject and object URIs for non-`type` triples from $\mathtt{G}$ should not appear in the summary. However, given that property URIs are often specified in SPARQL queries [1], and that typically there are far less distinct property URIs than the subject or object URIs [4], property URIs should be preserved by the summary. This leads us to considering the following SPARQL dialect:

Definition 3. *(RBGP queries) A* relational basic graph pattern query *(RBGP) is a conjunctive SPARQL query whose body has: (i) URIs in all the property positions, (ii) a URI in the object position of every* `type` *triple, and (iii) variables in any other positions.*

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

3 RDF Summaries

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

We introduce our summaries below; examples are delegated to [5] and can also be found at `https://team.inria.fr/oak/projects/rdfsummary/`.

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* `type`*) 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* `type`*)*

- **DT1** *If* $\mathtt{s}\ \mathtt{p}\ \mathtt{o},\ \mathtt{s}\ \mathtt{type}\ \mathtt{c}$ *are in* $\mathtt{G}$*, then* $\mathcal{S}(\mathtt{p})\ \mathtt{type}\ \mathtt{c}$ *is in* $\mathtt{B_G}$*;*
- **DT2** *if* $\mathtt{s}\ \mathtt{p}\ \mathtt{o},\ \mathtt{o}\ \mathtt{type}\ \mathtt{c}$ *are in* $\mathtt{G}$*, then* $\mathcal{T}(\mathtt{p})\ \mathtt{type}\ \mathtt{c}$ *is in* $\mathtt{B_G}$*;*
- **DT3** *Let* n_{all} *be set to* `newURI()`*. If* $\mathtt{s}\ \mathtt{type}\ \mathtt{c} \in \mathtt{G}$*, and* $\nexists \mathtt{s}\ \mathtt{p}\ \mathtt{o} \in \mathtt{G}$*, then* $n_{all}\ \mathtt{type}\ \mathtt{c} \in \mathtt{B_G}$.

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, 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 $\mathtt{R_G}$. For space reasons, the definition is delegated to [5]. 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.*

Summary properties Both summaries meet our requirements (*i*), (*iii*) and (*iv*) as follows. We say two summary graphs are *equivalent*, denoted $\equiv$, iff they are identical up to a bijection between their sets of URIs. The summaries *commute with saturation*, i.e., $(\mathtt{S_G})^\infty \equiv \mathtt{S_{G^\infty}}$, and are RBGP accurate. The $\mathtt{B_G}$ is fully RBGP representative, and the $\mathtt{R_G}$ is representative of *RBGPs having no more than one* `type` *triple with the same subject.* This follows from a *graph homomorphism* from $\mathtt{G}^\infty$ to $(\mathtt{S_G})^\infty$ [5]. Observe that $\mathtt{S_G}$ is not a core of $\mathtt{G}$, since we cannot guarantee a homomorphism from $\mathtt{S_G}$ to $\mathtt{G}$ ($\mathtt{S_G}$ may comprise false positives).

The size of the baseline summary is bounded by the size of $\mathtt{G}$'s schema plus the number of data properties and class assertions from $\mathtt{G}$. It can be built in $O(|\mathtt{G}|^2)$ time. Computing the refined summary has $O(|\mathtt{G}|^5)$ complexity, requiring an efficient underlying system e.g., based on triple partitioning and indexing or a distributed processing platform such as [2]. An upper bound for its size is the number of classes in $\mathtt{G}$ $\times$ the number of distinct data properties in $\mathtt{G}$.

Acknowledgments This work has been partially funded by the projects Datalyse "Investissement d'Avenir" and ODIN "DGA RAPID".

References

[1] 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.

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

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

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

[5] Technical report, 2015.