The VLDB Journal (2019) 28:295–327
https://doi.org/10.1007/s00778-018-0528-3

REGULAR PAPER

Summarizing semantic graphs: a survey

Šejla Čebirić[1] · François Goasdoué[2] · Haridimos Kondylakis[3] · Dimitris Kotzinos[4] · Ioana Manolescu[1] · Georgia Troullinou[3] · Mussab Zneika[4]

Received: 6 March 2018 / Revised: 1 November 2018 / Accepted: 16 November 2018 / Published online: 3 December 2018

Abstract
The explosion in the amount of the available RDF data has lead to the need to explore, query and understand such data sources. Due to the complex structure of RDF graphs and their heterogeneity, the exploration and understanding tasks are significantly harder than in relational databases, where the schema can serve as a first step toward understanding the structure. *Summarization* has been applied to RDF data to facilitate these tasks. Its purpose is to extract concise and meaningful information from RDF knowledge bases, representing their content as faithfully as possible. There is no single concept of RDF summary, and not a single but many approaches to build such summaries; each is better suited for some uses, and each presents specific challenges with respect to its construction. This survey is the first to provide a comprehensive survey of summarization method for semantic RDF graphs. We propose a taxonomy of existing works in this area, including also some closely related works developed prior to the adoption of RDF in the data management community; we present the concepts at the core of each approach and outline their main technical aspects and implementation. We hope the survey will help readers understand this scientifically rich area and identify the most pertinent summarization method for a variety of usage scenarios.

Keywords Semantic summaries · Summaries · Semantic graphs

1 Introduction

Semantic languages and models are increasingly used in order to describe, represent and exchange data in multiple domains and forms. In particular, given the prominence of the World Wide Web Consortium (W3C)[1] in the international technological arena, its standard model for representing semantic graphs, namely RDF, has been widely adopted. Many RDF Knowledge Bases (KBs, in short) of millions or even billions of triples are now shared through the Web, also thanks to the development of the Open Data movement, which has evolved jointly with the data linking best practices based on RDF. A famous repository of open RDF graphs is the Linked Open Data cloud, currently referencing more than 62 billion RDF triples, organized in large and complex RDF data graphs [87]. Further, several RDF graphs are conceptually linked together into one, as soon as a node identifier appears in several graphs. This enables querying KBs together, and increases the need to understand the basic properties of each data source before figuring out how they can be exploited together.

✉ Haridimos Kondylakis
kondylak@ics.forth.gr

Šejla Čebirić
sejla.cebiric@inria.fr

François Goasdoué
fg@irisa.fr

Dimitris Kotzinos
Dimitrios.Kotzinos@u-cergy.fr

Ioana Manolescu
ioana.manolescu@inria.fr

Georgia Troullinou
troulin@ics.forth.gr

Mussab Zneika
Mussab.Zneika@ensea.fr

1 Inria and LIX (UMR 7161, CNRS and Ecole polytechnique), Palaiseau, France

2 Univ Rennes, Inria, CNRS, IRISA, Rennes, France

3 Institute of Computer Science, FORTH, Heraklion, Greece

4 Lab. ETIS UMR 8051, University of Paris-Seine, University of Cergy-Pontoise, ENSEA, CNRS, 95000 Pontoise, France

[1] http://www.w3.org.

The fundamental difficulty toward understanding an RDF graph is its lack of a standard structure (or schema), as RDF graphs can be very heterogeneous and the basic RDF standard does not give means to constrain graph structure in any way. Ontologies can (but do not have to) be used in conjunction with RDF data graphs, in order to give them more meaning, notably by describing the possible classes resources may have, their properties, as well as relationships between these classes and properties. On the one hand, ontologies do provide an extra entry point into the data, as they allow to grasp its conceptual structure. On the other hand, they are sometimes absent, and when present, they can be themselves quite complex, growing up to hundreds or thousands of concepts; SNOMED-CT,[2] a large medical ontology, comprises millions of terms.

To cope with these layers of complexity, RDF graph summarization has aimed at extracting concise but meaningful overviews from RDF KBs, representing as close as possible the actual contents of the KB. RDF summarization has been used in multiple application scenarios, such as identifying the most important nodes, query answering and optimization, schema discovery from the data, or source selection, and graph visualization to get a quick understanding of the data. It should be noted that indexing, query optimization and query evaluation were studied as standalone problems in the data management areas, before the focus went to semantic RDF graphs; therefore, several summarization methods initially studied for data graphs were later adapted to RDF. Among the currently known RDF summarization approaches, some only consider the graph data without the ontology, some others consider only the ontology, finally some use a mix of the two. Summarization methods rely on a large variety of concepts and tools, comprising structural graph characteristics, statistics, pattern mining or a mix thereof. Summarization methods also differ in their usage scope. Some summarize an RDF graph into a smaller one, allowing some RDF processing (e.g., query answering) to be applied on the summary (also). The output of other summarization methods is a set of rules, or a set of frequent patterns, an ontology etc.

Summarizing semantic graphs is a multifaceted problem with many dimensions, and thus, many algorithms, methods and approaches have been developed to cope with it. As a result, there is now a confusion in the research community about the terminology in the area, further increased by the fact that certain terms are often used with different meanings in the relevant literature, denoting similar, but not identical research directions or concepts. We believe that this lack of terminology and classification hinders scientific development in this area.

To improve understanding of this field and to help students, researchers or practitioners seeking to identify the summarization algorithm, method or tool best suited for a specific problem, this survey attempts to provide a first systematic organization of knowledge in this area. We propose a taxonomy of RDF (and most representative, prior graph) summarization approaches. Then, we classify existing works according to the main class of algorithmic notions they are based on; further, for each work, we specify their accepted inputs, outputs, and when a tool is publicly available, we provide the reference to it. We place each of the works in the space defined by the dimensions of our classification; we summarize their main concepts and compare them when appropriate.

Since our focus is on RDF graph summarization techniques, we leave out of our scope graph summarization techniques tailored for other classes of graphs, e.g., biological data graphs [89], social networks [57]. We focus on techniques that have either been specifically devised for RDF, or adapted to the task of summarizing RDF graphs. The literature comprises surveys on generic (non-RDF) graph summarization, and/or partial surveys related to our area of study. The authors of [112] present generic graph summarization approaches, with a main focus on grouping-based methods. A recent survey [59] has a larger focus than ours. It considers static graphs as well as graphs changing over time; graphs which are just connection networks (node and edge labels are non-existent or ignored), but also labeled directed or undirected graphs, which can be seen as simple subsets of RDF. Also, a recent tutorial [40] covers a similar set of topics. However, given their broad scope, these works describe areas of work we are not concerned with, such as social (network) graph summarization, and ignore many of the proposals specifically tailored for RDF graphs, which are labeled, oriented, heterogeneous, and may be endowed with type information and semantics. In contrast, our survey seeks to answer a need we encountered among many Semantic Web practitioners, for a comprehensive review of summarization techniques tailored exactly to such graphs. Another recent work [79] focuses on metrics used for ontology summarization only, whereas we consider both RDF graphs and their ontologies.

Our survey is structured as follows: Sect. 2 recalls the foundations of the RDF data model, and RDF Schema (RDFS, in short), the simplest ontology language which can be used in conjunction with RDF to specify semantics for its data. Section 3 describes RDF summarization scope, applications and dimensions of analysis for this survey. In Sect. 4, we classify along these dimensions a selection of the main graph summarization works which preceded works on RDF summarization. We include a short discussion of these works here, as they were the first to introduce a set of concepts crucial for summarization, and on which RDF-specific summaries have built. In the sequel of the survey, Sects. 5, 6, 7 and 8 analyze the related works in each category. Finally,

[2] https://www.snomed.org/snomed-ct.

Sect. 9 concludes this paper and identifies fields of future exploration.

2 Preliminaries: RDF graphs

We recall here the core concepts and notations related to RDF graphs. At a first glance, these can be considered particular cases of labeled, oriented graphs, and indeed classical graph summarization techniques have been directly adapted to RDF; we recall them in Sect. 2.1. Then, we present RDF graphs in Sect. 2.2, where we introduce the terminology and specific constraints which make up the RDF standard, established by the W3C; we also introduce here ontologies, which play a central role in most RDF applications, with a focus on the simple RDF Schema ontology language. From a database perspective, the most common usage of RDF graphs is through queries; therefore, we recall the Basic Graph Pattern (BGP) dialect at the core of the SPARQL RDF query language in Sect. 2.3. Finally, a brief discussion of more expressive ontology languages, sometimes used in conjunction with RDF graphs, is provided in Sect. 2.4.

2.1 Labeled directed graphs: core concepts

Labeled directed graphs are the core concept allowing to model RDF datasets. Further, most (not all) proposals for summarizing an RDF graph also model the summary as a directed graph. Thus, without loss of generality, we will base our discussion on this model. Note that it can be easily generalized to more complex graphs, e.g., those with (multi-) labeled nodes.

Given a set A of labels, we denote by $G = (V, E)$ an A-edge labeled directed graph whose vertices are V, and whose edges are $E \subseteq V \times A \times V$.

Figure 1 displays two such graphs; A edge labels are attached to edges. Node labels will be used/explained shortly in our discussion.

In addition, the notions of *graph homomorphism* and *graph isomorphism* frequently appear in graph summary proposals:

Definition 1 (*Homomorphism and isomorphism*) Let $G = (V, E)$ and $G' = (V', E')$ be two A-edge-labeled directed graphs. A function $\phi : V \to V'$ is a homomorphism from G to G' iff for every edge $(v_1, l, v_2) \in E$ there is an edge $(\phi(v_1), l, \phi(v_2) \in G'$. If, moreover, ϕ is a bijection, and its inverse ϕ^{-1} is also a homomorphism from G' into G, then ϕ is an isomorphism.

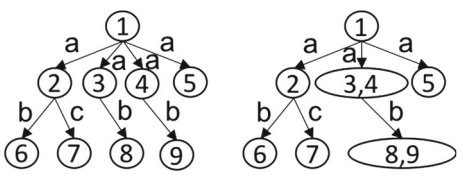

Fig. 1 Sample edge-labeled directed graphs

A homomorphism from G to G' ensures that the graph structure present in G has an "image" into G'. For our discussion, this is interesting in three different settings:

1. If G is a data graph and G' is a summary graph representing G, a homomorphism from G to G' ensures that every subgraph of G has an image in G'.
2. Conversely, a homomorphism from a summary graph G' into the data graph G means that all the graph structures present in the summary also appear in the data graph.
3. If Q is a graph *query*, e.g., expressed in SPARQL, and G is a *data* graph, e.g., an RDF graph, the answer to Q on G, denoted $Q(G)$, is exactly defined through the set of homomorphisms which may be established from G to G'. Together with the two items above, this leads to several interesting relationships between queries, data graphs and their summaries, in particular allowing to use the summary to gain some knowledge about $Q(G)$ without actually evaluating it.

Observe that while homomorphisms between a graph and its summary have useful properties, an isomorphism would defeat the purpose of summarization, as two isomorphic graphs would have the same size.

In Fig. 1, the graph shown on the right is homomorphic to that on the left. Indeed, a homomorphism maps each node from the graph at left into the right graph node whose label contains its number.

2.1.1 Notations: node and edge counts

Throughout this survey, unless otherwise specified, N denotes the number of nodes and M the number of edges of a directed graph input to some summarization approach.

2.2 The resource description framework (RDF)

Our study of graph summarization techniques is centrally motivated by their interest when summarizing RDF graphs. RDF is the standard data model promoted by the W3C for Semantic Web applications.

2.2.1 RDF graph

An *RDF graph* (in short a *graph*) is a set of *triples* of the form (s, p, o). A triple states that a *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 RDF specification [106],

Assertion	Triple	Relational notation
Class	$(\mathtt{s}, \text{rdf:type}, \mathtt{o})$	$\mathtt{o}(\mathtt{s})$
Property	$(\mathtt{s}, \mathtt{p}, \mathtt{o})$	$\mathtt{p}(\mathtt{s}, \mathtt{o})$

Constraint	Triple	OWA interpretation
Subclass	$(\mathtt{s}, \prec_{sc}, \mathtt{o})$	$\mathtt{s} \subseteq \mathtt{o}$
Subproperty	$(\mathtt{s}, \prec_{sp}, \mathtt{o})$	$\mathtt{s} \subseteq \mathtt{o}$
Domain typing	$(\mathtt{p}, \hookleftarrow_{d}, \mathtt{o})$	$\Pi_{\text{domain}}(\mathtt{s}) \subseteq \mathtt{o}$
Range typing	$(\mathtt{p}, \hookrightarrow_{r}, \mathtt{o})$	$\Pi_{\text{range}}(\mathtt{s}) \subseteq \mathtt{o}$

Fig. 2 RDF (top) and RDFS (bottom) statements

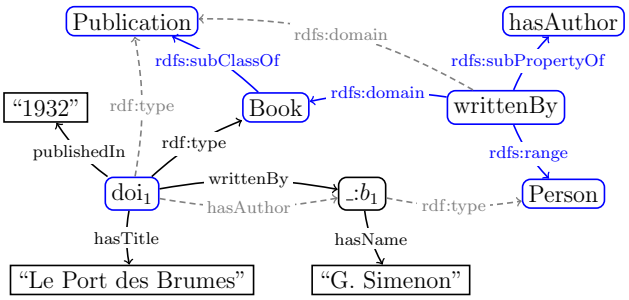

Fig. 3 RDF graph and its implicit triples

belonging to $(\mathcal{U} \cup \mathcal{B}) \times \mathcal{U} \times (\mathcal{U} \cup \mathcal{B} \cup \mathcal{L})$ where $\mathcal{U}$ is a set of Uniform Resource Identifiers (URIs), $\mathcal{L}$ a set of typed or untyped literals (constants), and $\mathcal{B}$ a set of blank nodes (unknown URIs or literals); $\mathcal{U}, \mathcal{B}, \mathcal{L}$ are pairwise disjoint. Blank nodes are essential features of RDF allowing to support *unknown URI/literal tokens*. These are conceptually similar to the labeled nulls or variables used in incomplete relational databases [1], as shown in [27]. As described above, it is easy to see that any RDF graph is a labeled graph as described in Sect. 2.1. However, as we explain below, RDF graphs may contain an *ontology*, that is, a set of graph edges to which standard ontology languages attach a special interpretation. The presence of ontologies raises specific challenges when summarizing RDF graphs, which do not occur when only plain data graphs are considered.

2.2.2 Notations

We use s, p, and o as placeholders for subjects, properties and objects, respectively. Literals are shown as strings between quotes, e.g., "*string*". Figure 2 (top) shows how to use triples to describe resources, that is, to express class (unary relation) and property (binary relation) assertions. The RDF standard [106] has 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.

Example 1 (RDF graph) For example, the following RDF graph G describes a book, identified by doi_1, its author (a blank node $\_{:}b_1$ whose name is known), title and date of publication:

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

2.2.3 RDF schema (RDFS)

RDFS allows enhancing the assertions made in an RDF graph with the use of an ontology, i.e., by declaring *semantic constraints* between the classes and the properties they use. Figure 2 (bottom) shows the four main kinds of RDFS constraints, and how to express them through triples hence particular graph edges. For concision, we denote the properties expressing *subclass*, *subproperty*, *domain* and *range* constraints by the symbols $\prec_{sc}$, $\prec_{sp}$, $\hookleftarrow_{d}$ and $\hookrightarrow_{r}$, respectively. Here, "domain" denotes the first, and "range" the second attribute of every property.

The RDFS constraints depicted in Fig. 2 are interpreted under the *open-world assumption (OWA)* [1], i.e., as *deductive* constraints. For instance, if the triple (hasFriend, $\hookleftarrow_{d}$, Person) and the triple (Anne, hasFriend, Marie) hold in the graph, then so does the triple (Anne, τ, Person). The latter is due to the domain constraint in Fig. 2.

Example 2 (RDF graph with an RDFS ontology) Assume that the RDF graph G in the preceding example is extended with the RDFS ontological constraints: (Book, $\prec_{sc}$, Publication), (writtenBy, $\prec_{sp}$, hasAuthor), (writtenBy, $\hookleftarrow_{d}$, Book) and (writtenBy, $\hookrightarrow_{r}$, Person). The resulting graph is depicted in Fig. 3. Its implicit triples are those represented by dashed-line edges.

2.2.4 RDF entailment

An important feature of RDF graphs are *implicit triples*. Crucially, these are considered part of the RDF graph even though they are not explicitly present in it, e.g., the dashed-line G edges in Fig. 3, hence require attention for RDF graph summarization.

W3C names *RDF entailment* the mechanism through which, based on a set of explicit triples and some *entailment rules*, implicit RDF triples are derived. We denote by $\vdash^{i}_{\text{RDF}}$ *immediate entailment*, i.e., the process of deriving new triples through a *single* application of an entailment rule. More generally, a triple $(\mathtt{s}, \mathtt{p}, \mathtt{o})$ is entailed by a graph G, denoted $G \vdash_{\text{RDF}} (\mathtt{s}, \mathtt{p}, \mathtt{o})$, if and only if there is a sequence

of applications of immediate entailment rules that leads from G to $(\mathtt{s}, \mathtt{p}, \mathtt{o})$ (where at each step, triples previously entailed are also taken into account).

2.2.5 Saturation

The immediate entailment rules allow defining the finite *saturation* (a.k.a. closure) of an RDF graph G, which is the RDF graph G^∞ defined as the fixed-point obtained by repeatedly applying $\vdash^i_{\text{RDF}}$ 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). An obvious connection holds between the triples entailed by a graph G and its saturation: $G \vdash_{\text{RDF}} (\mathtt{s}, \mathtt{p}, \mathtt{o})$ if and only if $(\mathtt{s}, \mathtt{p}, \mathtt{o}) \in G^\infty$.

RDF entailment is part of the RDF standard itself; in particular, *the answers to a query posed on G must take into account all triples in G^∞* [109], since in the presence of RDF Schema constraints, *the semantics of an RDF graph is its saturation* [106]. As a result, the summarization of an RDF graph should reflect its saturation, e.g., by summarizing the saturation of the graph instead of the graph itself.

Example 3 (RDF entailment and saturation) The saturation of the RDF graph comprising RDFS constraints G, displayed in Fig. 3, is the graph G^∞ obtained by adding to G all its implicit triples that can be derived through RDF entailment, i.e., the graph G in which the implicit/dashed edges are made explicit/solid ones.

We introduce below a few more notions we will need in order to describe existing RDF summarization proposals.

2.2.6 Instance and schema graph

An RDF instance graph is made of assertions only (recall Fig. 2), while an RDF schema graph is made of constraints only (i.e., it is an ontology). Further, an RDF graph can be partitioned into its (disjoint) instance and schema subgraphs.

2.2.7 Properties and attributes of an RDF graph

While this is not part of the W3C standard, some authors use *attribute* to denote a property (other than those built in the RDF and RDFS standards, such as τ, $\hookleftarrow_d$ etc.) of an RDF resource such that the property value is a literal. In these works, the term *property* is reserved for those RDF properties whose value is an URI.

Example 4 (Instance, schema, properties and attributes of an RDF graph) The RDF graph G shown in Fig. 3 consists of the RDF schema graph comprising the blue triples, and of the RDF instance graph comprising the black triples. Further, within this G instance subgraph, the properties considered attributes are the following: publishedIn, hasTitle and hasName.

2.3 BGP queries

SPARQL[3] is the standard W3C query language used to query RDF graphs. We consider its popular conjunctive fragment consisting of *Basic Graph Pattern* (BGP) queries. Subject of several recent works [9,26,27,77,94], BGP queries are also the most widely used in real-world applications [53,77]. A BGP is a generalization of an RDF graph in which variables may also appear as subject, property and object of triples.

2.3.1 Notations

In the following we use the conjunctive query notation $Q(\bar{x})$:- $t_1, \ldots, t_\alpha$, where $\{t_1, \ldots, t_\alpha\}$ is a BGP. The head of Q is $Q(\bar{x})$, and the body of Q is $t_1, \ldots, t_\alpha$. The query head variables $\bar{x}$ are called *distinguished variables*, and are a subset of the variables occurring in $t_1, \ldots, t_\alpha$; for boolean queries $\bar{x}$ is empty. We denote by $\mathtt{VarBl}(Q)$ the set of variables *and* blank nodes occurring in the query Q. In the sequel, we will use x, y, z, etc. to denote variables in queries.

2.3.2 Query evaluation

Given a query $Q(\bar{x})$:- $t_1, \ldots, t_\alpha$ and an RDF graph G, the *evaluation of Q against G* is:

$$Q(G) = \{\Phi(\bar{x}) \mid \Phi : \mathtt{VarBl}(Q) \to \mathtt{Val}(G) \text{ is a } Q \text{ to } G \text{ homomorphism such that } \{\Phi(t_1), \ldots, \Phi(t_\alpha)\} \subseteq G\}$$

where we denote by $\Phi(t)$ (resp. $\Phi(\bar{x})$) the result of replacing every occurrence of a variable or blank node $e \in \mathtt{VarBl}(Q)$ in the triple t (resp. the distinguished variables $\bar{x}$), by the value $\Phi(e) \in \mathtt{Val}(G)$.

2.3.3 Query answering

The evaluation of Q against G uses only G's explicit triples, thus may lead to an incomplete answer set. The (complete) *answer set* of Q against G is obtained by the evaluation of Q against G^∞, denoted by $Q(G^\infty)$.

Example 5 (Query evaluation versus answering) The query below asks for the author's name of "Le Port des Brumes":

$$Q(x_3) \text{:- } (x_1, \mathtt{hasAuthor}, x_2),\ (x_2, \mathtt{hasName}, x_3)\ (x_1, \mathtt{hasTitle}, \text{"}\mathtt{Le\ Port\ des\ Brumes}\text{"})$$

[3] https://www.w3.org/TR/rdf-sparql-query/.

Its answer against the explicit and implicit triples of our sample graph is: $Q(G^{\infty}) = \{\langle$"G. Simenon"$\rangle\}$.

Note that evaluating Q only against G leads to the empty answer, which is obviously incomplete.

2.4 OWL

Semantic graphs considered in the literature for summarization sometimes go beyond the expressiveness of RDF, which comes with the simple RDF Schema ontology language. The standard by W3C for semantic graphs is the OWL [107,108] family of dialects that builds on Description Logics (DLs) [4].

DLs are first-order logic languages that allow describing a certain application domain by means of *concepts*, denoting sets of objects, and *roles*, denoting binary relations between concept instances. DL *dialects* differ in the ontological constraints they allow expressing on complex concepts and roles, i.e., defined by DL formula. One of the most important issues in DLs is the trade-off between expressive power and computational complexity of reasoning with the constraints (consistency checking, query answering, etc.)

The first flavor of OWL [107] consists of three dialects of increasing complexity: OWL-Lite, OWL-DL and OWL-Full. Unfortunately, very basic reasoning (concept satisfiability) in these dialects is highly intractable: ExpTime-complete in OWL-lite that corresponds to the $\mathcal{SHIF}_{\mathcal{D}}$ DL, NExpTime-complete in OWL-DL that corresponds to the $\mathcal{SHIO}\mathcal{N}_{\mathcal{D}}$ DL, and even undecidable in OWL-full. A second flavor of OWL [108], a.k.a. OWL2, defines three new dialects, OWL2 EL based on the $\mathcal{EL}$ DL, OWL2 QL based on the DL-lite$_{\mathcal{R}}$ DL and OWL2 RL which can be expressed using logical rules. These new dialects comes with PTIME complexity for most of the reasoning tasks. In particular, data management tasks (consistency checking, query answering, etc.) under OWL2 QL/DL-lite$_{\mathcal{R}}$ ontologies have the same complexity as their counterparts in the relational database model [10].

3 RDF summarization: scope, applications and dimensions of analysis for this survey

As we shall see, RDF summarization has been attached many different meanings in the literature, and research is still ongoing. Therefore, we start with delimiting the scope of RDF summarization as considered in this survey (Sect. 3.1), before describing the RDF summary applications most frequently encountered within this scope (Sect. 3.2), and finally presenting several dimensions along which the corresponding RDF summarization techniques can be classified (Sect. 3.3).

3.1 Scope

Our goal in this survey is to study summarization notions and tools which are useful to concrete RDF data management applications. We will thus discuss a broad set of techniques, some of which are also used outside our target RDF data management contexts. However, to keep the survey focused, self-contained, and useful to RDF practitioners, we do not cover graph summarization or clustering techniques designed for very specific classes of graphs. For instance, while social network graphs can be modeled in RDF, such graphs have a very specific semantics, for instance, to reflect the important role of "user" nodes. Instead, we aim to cover summarization of general RDF graphs (without making assumptions on their application domain), without ontologies (in which case they basically coincide with labeled oriented graphs) or with ontologies (that are a specific, crucial feature of RDF data graphs).

Our review of the literature leads us to the following generic definition. An RDF summary is one or both among the following:

1. A *compact information*, extracted from the original RDF graph; intuitively, summarization is a way to extract meaning from data while reducing its size;
2. A *graph*, which some applications can *exploit instead of the original RDF graph*, to perform some tasks more efficiently; in this vision, a summary represents (or stands for) the graph in specific settings.

Clearly, these notions intersect, e.g., many graph summaries extracted from the RDF graphs are compact and can be used for instance to make some query optimization decisions; these fit into both categories. However, some RDF summaries are not graphs; some (graph or non-graph) summaries are not always very compact, yet they can be very useful etc.

3.2 Applications

We illustrate the above generic definition of an RDF summary through a (non-exhaustive) list of **uses and applications**.

3.2.1 Indexing

Most (RDF) summarization methods from the literature build summaries which are smaller graphs; each summary node represents several nodes of the original graph G. This smaller graph, then, serves as an index as follows. The identifiers of all the G nodes represented by each summary node v are associated with the node v. To process a query on G, we firstly identify the summary nodes, which may match the query; then identify based on the index, the graph nodes

corresponding to these summary nodes, as a first step toward answering the query.

3.2.2 Estimating the size of query results

Consider a summary defined as a set of statistics about property (edge label) co-occurrence in G, that is: the summary stores, for any two properties a, b appearing in G, the number of nodes which have at least an outgoing a edge and at least an outgoing b edge. If a query searches, e.g., for resources having both a "description" and an "endorsement" in an RDF graph storing product information, if the summary indicates that there are no such resources, we can return an empty query answer without consulting G. Further, assume that a BGP query requires a resource with properties p_1, p_2, somehow connected to another resource with properties p_3, p_4. If the summary shows that the former property combination is much rarer than the latter, a query optimizer can exploit this to start evaluating the query from the most selective conditions p_1, p_2.

3.2.3 Making BGPs more specific

BGPs queries may comprise path expression with wildcards; these are hard to evaluate, as they require traversing a potentially large part of G. A graph summary may help understand, e.g., that a path specified as "any number of a edges followed by one or more b edges" corresponds to exactly two data paths in G, namely: a b edge; and an a edge followed by a b edge. These two short and simple path queries are typically evaluated very efficiently.

3.2.4 Source selection

One can detect based on a summary whether a graph is *likely* to have a certain kind of information that the user is looking for, without actually consulting the graph. In a distributed query processing setting, this can be used to know which data partition(s) are helpful for a query; in a LOD cloud querying context, when answering queries over a large set of initially unknown data sources, this problem is typically referred to as source selection.

3.2.5 Graph visualization

A graph-shaped summary may be used to support the users' discovery and exploration of an RDF graph, helping them get acquainted with the data and/or as a support for visual querying.

3.2.6 Vocabulary usage analysis

RDF is often used as a mean to standardize the description of data from a certain application domain, e.g., life sciences, Web content metadata etc. A standardization committee typically works to design a vocabulary (or set of standard data properties) and/or an ontology; application designers learn the vocabulary and ontology and describe their data based on them. A few years down the road, the standard designers are interested to know which properties and ontology features were used, and which were not; this can inform decisions about future versions of the standard.[4]

3.2.7 Schema (or ontology) discovery

When an ontology is not present in an RDF graph, some works aim at extracting it from the graph. In this case, the summary is meant to be used as a schema, which is considered to have been missing from the initial data graph.

3.3 Classification of RDF summarization methods

From a scientific viewpoint, existing summarization proposals are most meaningfully classified according to the main algorithmic idea behind the summarization method:

1. *Structural methods* are those which consider first and foremost the graph structure, respectively the paths and subgraphs one encounters in the RDF graph. Given the prominence of applications and graph uses, where structural conditions are paramount, graph structure is prominently used in summarization techniques.
   - *Quotient* A particular natural concept when building summaries is that of quotient graphs (Definition 2). They allow characterizing some graph nodes as "equivalent" in a certain way, and then summarizing a graph by assigning a representative to each class of equivalence of the nodes in the original graph. A particular feature of structural quotient methods is that each graph node is represented by exactly one summary node, given that one node can only belong to one equivalence class.
   - *Non-quotient* Other methods for structurally summarizing RDF graphs are based on other measures, such as centrality, to identify the most important nodes, and interconnect them in the summary. Such methods aim at building an overview of the graph, even if (unlike quotient summaries) some graph nodes may not be represented at all.

[4] Thanks to William van Voensel from schema.org for sharing this application with us.

2. *Pattern-mining methods* These methods employ mining techniques for *discovering* patterns in the data; the summary is then built out of the patterns identified by mining.
3. *Statistical methods* These methods summarize the contents of a graph quantitatively. The focus is on counting occurrences, such as counting class instances or building value histograms per class, property and value type; other quantitative measures are frequency of usage of certain properties, vocabularies, average length of string literals etc. Statistical approaches may also explore (typically small) graph patterns, but always from a quantitative, frequency-based perspective.
4. *Hybrid methods* To this category belong works that combine structural, statistical and pattern-mining techniques.

Another interesting dimension of analysis is the required **input** by each summarization method, in terms of the actual dataset, and of other user inputs which some methods need:

1. *Input parameters* Many works in the area require user parameters to be defined, e.g., user-specified equivalence relations, maximum summary size, weights assigned to some graph elements etc., whereas others are completely user independent. While parameterized methods are able to produce better results in specific scenarios, they require some understanding of the methodology and as such limit their exploitation ability only to experts.
2. *Input Dataset* Different works have different requirements from the dataset they get as input. RDF data graphs are most frequently accepted, usually RDF/S and/or OWL are used for specifying graph semantics, whereas only very few works consider DL models. In addition, some works consider or require only ontologies (semantic schema), whereas other works exploit only instances. Hybrid approaches exploit both instance and schema information. For instance, the instance and schema information can be used to compute the summary of the saturated graph, even if the instance graph is not saturated.

For what concerns the **summarization output**, we identify the following dimensions:

1. *Type* This dimension differentiates techniques according to the nature of the final result (summary) that is produced. The summary is sometimes a graph, while in other cases it may be just a selection of frequent structures such as nodes, paths, rules or queries.
2. *Nature* Along this dimension, we distinguish summaries which only output instance representatives, from those that output some form of summary a posteriori schema, and from those that output both.

Availability Last but not least, from a practical perspective, it is interesting to know the availability of a given summarization service. This will allow a direct comparison with future similar tools:

1. *System/tool* Several summarization approaches are made available by their authors as a tool or system shared with the public; in our survey, we signal when this is the case. In addition, some of the summarization tools can be readily tested from an online deployment provided by the authors.
2. *Open source* The implementation of some summarization methods is provided in open source by the authors, facilitating comparison and reuse.

We also consider the quality characteristics of each individual algorithm. Quality has to do with: completeness in terms of coverage, precision and recall of the results if an "ideal" summary is available as a gold standard to compare with, the connectivity of the computed summary and, at the end, computational complexity. Given variety of RDF summarization approaches, it is not easy to define and evaluate a single meaningful notion of quality. A more comprehensive effort to establish a generic framework for computing quality metrics on summaries is proposed in [121], where authors discuss summarization quality concerning both the schema and instances levels. However, difficulties remain, e.g., identifying the complexity of a summarization algorithm is not always possible when the available description of the algorithm does not provide sufficient information.

The main categorization we retain for the different RDF summarization approaches is based on their measurable/identifiable characteristics and not on their intended use. This is because the boundaries among the different usages are not very clear and there are types of methods that can be used in diverse cases/applications. Thus, the advantages and/or disadvantages for each category of methods cannot be identified in a generic way; however, we inserted a discussion whenever pertinent.

Figure 4 depicts a high-level taxonomy of the RDF summarization works, based on the aforementioned dimensions. Note that many of the dimensions are orthogonal; thus, a work may be classified in multiple categories. In the sequel, we classify the works, describe the main ideas and the implemented algorithms. Then, we identify the specific dimension of analysis captured in Fig. 4 for each of these works.

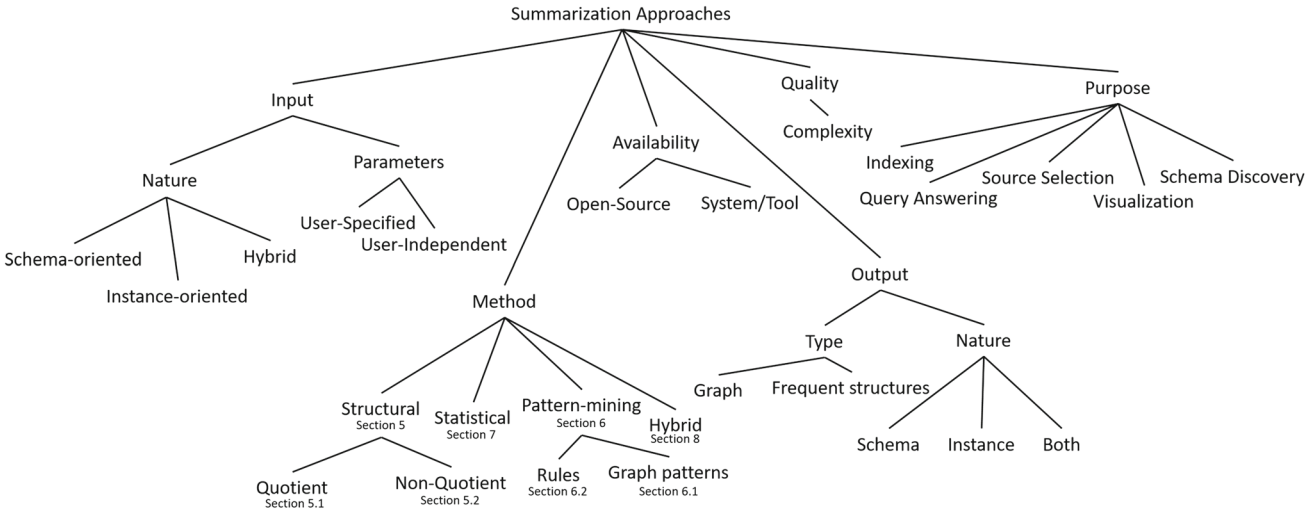

Fig. 4 A taxonomy of the works in the area

4 Generic graph (non-RDF) summarization approaches

In this section, we review generic graph summarization approaches. While these have not been specifically devised for RDF, they have either been applied to RDF subsequently, or served as inspiration for similar RDF-specific proposals. An overview of the generic graph summarization works is provided in Tables 1 and 2. More precisely, Sect. 4.1 presents structural graph summarization methods, Sect. 4.2 describes works based on mining and statistics, while Sect. 4.3 considers summaries based on statistic and hybrid (structural and statistic) methods.

4.1 Structural graph summarization

The structural complexity and heterogeneity of graphs make query processing a challenging problem, due to the fact that nodes which may be part of the result of a given query can be found anywhere in the input graph. To tame this complexity and direct query evaluation to the right subsets of the data, graph summaries have been proposed as a basis for indexing, by storing next to each summary node, the IDs of the original graph nodes summarized by this node; this set is typically called the *extent*. Given a query, evaluating the query on the summary and then using the summary node extents allows obtaining the final query results.

4.1.1 Quotient graph summaries

Many proposals for indexing graph data are based on establishing some notion of equivalence among graph nodes, and storing the IDs of all nodes as the extent of the summary node. Formally, these correspond to quotient graphs, whose definition we recall below:

Definition 2 (*Quotient graph*) Let $G = (V, E)$ be an A-edge-labeled directed graph and $\equiv \subseteq V \times V$ be an *equivalence relation* over the nodes of V. The *quotient graph of* G *using* $\equiv$, denoted $G_{/\equiv}$, is an A edge-labeled directed graph having:

- a node u_S for each set S of $\equiv$-equivalent V nodes;
- an edge (v_{S_1}, l, v_{S_2}) iff there exists an E edge (v_1, l, v_2) such that v_{S_1} (resp. v_{S_2}) represents the set of V nodes $\equiv$-equivalent to v_1 (resp. v_2).

Prominent node equivalence relations used for graph summarization build on backward, forward, or backward-and-forward bisimulation [32]:

Definition 3 (*Backward bisimulation*) In an edge-labeled directed graph, a relation $\approx_b$ between the graph nodes is a **backward bisimulation** if and only if for any $u, v, u', v' \in V$:

1. If $v \approx_b v'$ and v has no incoming edge, then v' has no incoming edge;
2. If $v \approx_b v'$ and v' has no incoming edge, then v has no incoming edge;
3. If $v \approx_b v'$, then for any edge $u \xrightarrow{a} v$ there exists an edge $u' \xrightarrow{a} v'$ such that $u \approx_b u'$;
4. If $v \approx_b v'$, then for any edge $u' \xrightarrow{a} v'$ there exists an edge $u \xrightarrow{a} v$ such that such that $u \approx_b u'$.

Forward bisimulation, noted $\approx_f$, is defined similarly to backward simulation, but considers the outgoing edges of v and v', instead of the incoming ones. *Forward and backward*

Table 1 Graph summaries based on structural quotients

Work	Input requirements	Purpose	Output type	Output nature	System–theory
SNAP/k-SNAP [96,97]	Required user parameters	Visualization	Graph	Instance	Theory
Zhang et al. [114]	Required user parameters	Visualization	Graph	Instance	Theory
Louati et al. [60]	Required user parameters	Visualization	Graph	Instance	Theory
Fan et al. [22]	Required user-selected queries	Query answering	Graph	Node-labeled directed graph	Theory
DescribeX [18]	None	Query answering	Single root node-labeled graphs	Instance	System
Luo et al. [62]	None	Indexing, query answering	Graph	Instance	Theory
T-index [64]	Parametrized user input	Query answering	Multi-roots edge-labeled graphs	Instance	Theory
D(K)-index [82]	Parametrized user input	Query answering	Single root node-labeled graphs	Instance	Theory
F&B-index [37]	Parametrized user input	Query answering	Single root node-labeled graphs	Instance	Theory

simulation, noted $\approx_{fb}$, is both a backward and a forward bisimulation.

We already pointed out that, in Fig. 1, the graph on the right is homomorphic to that on the left. The former is actually the quotient graph of the latter using $\approx_{fb}$. In particular, the classes of $\approx_b$-equivalent nodes are $\{1\}, \{2, 3, 4, 5\}, \{6, 8, 9\}, \{7\}$, those of $\approx_f$-equivalent nodes are $\{1\}, \{2\}, \{3, 4\}, \{5, 6, 7, 8, 9\}$, and those of $\approx_{fb}$-equivalent nodes are $\{1\}, \{2\}, \{3, 4\}, \{5\}, \{6\}, \{7\}, \{8, 9\}$.

Finally, we remark that it easily follows from the bisimulation definitions that if $v \approx v'$, for instance for forward bisimulation, then any label path that can be followed from v in the graph G can also be followed from v' in G and the other way around. In other words, the same paths start (respectively, end) in two $\approx$-equivalent nodes. This condition is hard to meet in graphs that exhibit some structural heterogeneity: in such cases, every node is $\approx$-equivalent to very few (if any) other nodes.

The Template Index (or T-index) [64] summary considers that two graph nodes are equivalent if they are backward bisimilar. In particular, in a T-index, nodes represented together need to be reachable by the exact same set of incoming paths. The goal of the T-index is to speed up the evaluation of complex queries of a certain form (or template), such as $P.v$ (all nodes v reachable by a path matching the regular expression P) or $v.P.u$ (all v, u node pairs connected by a path matching P); the proposal generalizes to arbitrary arity queries, although the authors note it is likely to be most useful in the above two simplest forms. The simulation relation between the nodes of a graph G is known to be computable in $O(M * \log(M))$ [71] or $O(N * M)$ [32]; the cost drops to linear for acyclic graphs. All these algorithms assume the graph fits in memory.

To support efficient processing of graph queries that navigate both forward (in the direction of the graph edges) and backward, Kaushik et al. [37] describes the Forward and Backward Index (F&B) which considers two nodes equivalent if they are undistinguishable by any navigation path composed only of forward and backward steps (see Fig. 5 for an example). While this equivalence condition is very powerful, it is rarely satisfied by two nodes, thus the F&B-index is likely to have a large amount of nodes (close to the number of nodes in the original graph), making the manipulation of the F&B-index structure inefficient. To address the problem, the authors note that in practice not *any* path query is frequently asked by applications; therefore, it suffices to consider F&B equivalence of nodes as being undistinguishable by forward and backward navigation along the paths from a certain set only.

Another method proposed in [38,82], in order to make the F&B-index smaller and more manageable, is to consider bisimilarity restricted only to paths of a certain length around the graph nodes (see Fig. 5 for an example). This increases

Table 2 Other graph summary proposals

Work	Method	Input requirements	Purpose	Output type	Output nature	System–theory
Dataguide [28]	Structural non-quotient	None	Indexing, query answering	Single root, node- and edge-labeled graphs	Instance	System
Rudolf et al. [85]	Structural non-quotient	Required user parameters	Visualization	Property graph	Instance	System
Chen et al. Graph OLAP [16]	Structural non-quotient	None	OLAP	Multiple non-RDF graphs	Multiple Instances	Theory
Zhao et al. [117]	Pattern mining	None	Indexing, graph containment queries	Set of frequent trees	Instance	Theory
Yan et al. [111]	Pattern mining	Parameterized user input	Indexing, query answering	Tree	Instance	Theory
Koutra et al. [51]	Clustering, pattern mining	None	Visualization	Graph	Instance	Theory
Khan et al. [41]	Structural non-quotient, data mining	Required degree threshold, overlap ratio	Visualization	Graph	Instance	Theory
Navlakha et al. [69]	Structural non-quotient, data mining	Optional user-specified bounded-error parameter	Visualization	Graph	Instance	Theory
LeFevre et al. [55], Riondato et al. [83]	Structural non-quotient, data mining	Required number of summary nodes, size of the extent of summary nodes	Answering adjacency, degree and centrality queries	Graph	Instance	Theory
Chen et al. Randomized summaries [15]	Structural non-quotient, pattern mining	Required support threshold	Visualization	Graph	Instance	Theory

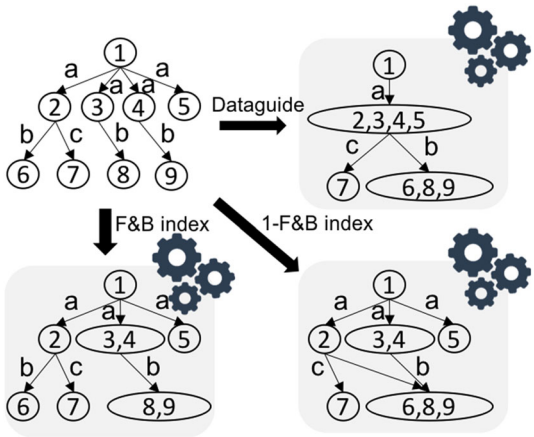

Fig. 5 Structural graph summarization examples

the chances that two nodes be considered equivalent, thus reducing the size of the bisimilarity-based summary. Limited bisimulation summaries like this, can be computed by evaluating structural group-by queries (for k-bounded bisimilarity), respectively, queries derived from the workload of interest (for workload-driven F&B summarization). While the theoretical complexity of such queries is $O(M^k)$, with M the number of edges and k the size of the most complex query involved, efficient graph query processors, with the help of good indexing, achieve much better performance in practice.

Consens et al. [18] considers the summarization of large Web document collections. While the main structure in this case consists of trees, they may also feature reference edges which turn the dataset into a global graph. The authors build a summary as a collection of regular expression queries, such that the set of results to these queries, together, make up a partition over the set of nodes in all the documents of the input collection. To each such regular expression is associated a set of cardinality statistics, to help application designers chose meaningful queries and inform them on the expected performance which may be reached on those queries. The dominant-cost operation required by this approach is computing simulations among N nodes; summaries can then be refined based on user-specified path queries.

Luo et al. [62] provides an I/O efficient external memory-based algorithm for constructing the k-bisimulation summary of a disk-resident graph on a single machine, based on several passes of sorting and gradually refining partitions of nodes on disk. The I/O complexity of the algorithm is $O(k * sort(M_p) + k * scan(N_p) + sort(N_p))$, where M_p, respectively, N_p are the numbers of disk pages required to store the graph edges, respectively, graph nodes, while $sort(\cdot)$ and $scan(\cdot)$ quantify the cost of an external sort, respectively, the cost to scan a certain number of pages.

4.1.2 Non-quotient graph summaries

There are also many methods that construct non-quotient graph summaries. They distinguish nodes according to several criteria/measures and create summaries in which summary nodes represent multiple nodes out of the original graph.

A Dataguide [28] (see Fig. 5 for an example) is a summary of a directed acyclic graph, having one node for each data path in the original graph. A graph node reachable by a set of paths belongs to the extents of all the respective Dataguide nodes. Thus, given a path query which may also contain wildcards, and a Dataguide, it is easy to identify the Dataguide nodes corresponding to the query, and from there their extents. The authors show how to integrate the Dataguide path index with more conventional indexes, e.g., a value index which gives access to all the nodes containing a certain constant value etc. A Dataguide is not a quotient: an input node may be represented by two Dataguide nodes, if it is reachable by two distinct paths in the input graph. Building a Dataguide out of a data graph amounts to determinizing an undeterministic finite automaton; the worst-case complexity of this is known to be exponential in the size of the input graph, yet only linear when the database is tree-structured.

Summary-based query answering with an acceptable level of error is considered in [41,69]. The focus is on graph compression while preserving bounded-error query answering and/or the ability to fully reconstruct the graph from the summary with the help of so-called "corrections" (i.e., edges to add to or remove from the "expansion" of the summary into the regular part of the graph it derives from). The nodes of the resulting structural summary represent partitions of similar nodes from G, while a summary edge exists between two summary nodes u and v only if the nodes from G represented by u and v are densely connected. Similarly, LeFevre and Terzi [55] aims to compress G, but without considering corrections, while their edge labels represent the number of edges within each partition set, and the number of edges between every two such sets. To determine similar nodes, Khan et al. [41] relies on locality-sensitive hashing, while [55,69] use a clustering method where pairs of nodes to merge are chosen based on the optimal value of an objective function. In [83], the authors build on the concepts of [55] to show how to obtain in polynomial time, summaries which are close to the optimal one (in terms of corrections needed. However, these works focus on node connectivity, and ignore the node and edge labels which crucially encode the data content of an RDF graph.

Tian et al. [96] proposes the SNAP (Summarization on Grouping Nodes on Attributes and Pairwise Relationships) technique, whose purpose is to construct, with some user input, a summary graph that can be used for visualization. A SNAP summary represents all the input graph nodes and edges: its nodes forms a partition of the input graph nodes, and there is an edge of type t between two summary nodes A and B if and only if some input graph node represented by A is connected through an edge of type t to an input graph node represented by B. Further, a SNAP summary has a minimal number of nodes such that (i) all input graph nodes represented by a summary node have same values for some user-selected attributes and (ii) every input graph node represented by some summary node is connected to some input graph nodes through edges of some user-selected types.

Fan et al. [22] considers reachability and graph pattern queries on labeled graphs, and builds *answer-preserving summaries* for such queries, that is: for a given graph G, summary $S(G)$ and query Q, there exists a query Q' which can be computed from Q, and a post-processing procedure P such that $P(Q'(S(G))) = Q(G)$. In other words, evaluating Q' on the summary and then applying the post-processing P leads to the result $Q(G)$. This property is rather strong; however, it is attained not under the usual query semantics based on graph homomorphism, underlying SPARQL, but under a *bounded graph simulation* one. Under these semantics, answering a query becomes P (instead of NP), at the price of not preserving the query structure (i.e., joins). The authors propose two parallel graph compression strategies, targeting different kinds of queries: (i) reachability queries, where we seek to know if one node is reachable from another; (ii) graph pattern queries, for which they attain a significant compression ratio. It should be again stressed that the authors consider these queries under non-standard, more lenient semantics than the ones used for RDF querying. The complexity of building their summaries are $O(N * M)$ for reachability queries, and $O(N * \log(M))$ for graph pattern queries.

4.2 Mining-based graph summarization

OLAP and data mining techniques applied to data graphs have considered them through global, aggregated views, looking for statistics and/or trends in the graph structure and content.

Yan et al. [111] leverages pattern-mining techniques to build graph indices in order to help processing a graph query. It firstly applies a frequent pattern-mining algorithm to iden-

tify all the frequent patterns with the size support constraint. Once the frequent patterns are extracted, they are organized in a prefix tree structure, where each pattern is associated with a list of ids of the graphs containing it. This prefix tree is then used to answer queries asking for the respective part of the graph. This approach by design does not reflect all data and is based on numeric information. [117] uses the same idea, but considers trees instead of graphs.

The *Vocabulary-based summarization of Graphs* (VoG) [51] aims at summarizing a graph by its characteristic subgraphs of some fixed types, which have been observed encoding meaningful information in real graphs: cliques, bi-partite cores, stars, chains, and approximations thereof. VoG first decomposes the input graph, using any clustering method, into possibly overlapping subgraphs, the (approximate) type of each is then identified using the Minimum Description Length principle [84]. Finally, the input graph summary is composed of some non-redundant subgraphs, picked by some heuristic like top k, hence may not reflect all the input graph. Importantly, the VoG method has been shown to scale well; it is near-linear in the number of edges. The VoG code is available for download.[5]

An aggregation framework for OLAP operations on labeled graphs is introduced in [16]. The authors assume as available an OLAP-style set of dimensions with their hierarchies and measures; in particular, graph topological information is used as aggregation dimensions. Based on these, they define a "graph cube" and investigate efficient methods for computing it. With a different perspective, [15] focuses on building out of node- and edge-labeled graphs, a *set* of *randomized summaries*, so that one can apply data mining techniques on the summary set instead of the original graph. Using the summary set leads to better performance, while guaranteeing upper bounds on the information loss incurred.

A graph summarization approach, based solely on the graph structure is reported in [58]. It produces a summary graph that describes the underlying topology characteristics of the original one. Every summary node, or supernode, comprises of a set of nodes from the original graph; every summary edge, or super-edge, represents an all-to-all connections between the nodes in the corresponding supernodes. The goal of this work is to generate a summary that minimizes the false positives and negatives introduced by this summarization. The authors investigate different distributed graph summarization methods, which proceed in an incremental fashion, gradually merging nodes into supernodes; the methods differ in the way they chose the pairs of nodes to be merged, and cut different trade-offs between efficiency (running time) and effectiveness (keeping the false positives and negatives under control). The method termed Dist-LSH selects node pairs with a high probability to be merged; the probability is estimated based on locality-sensitive hashing (LSH) of the nodes. The algorithms are implemented on top of the Apache Giraph framework.

Lin et al. [57] surveys many other quantitative, mining-oriented graph sampling and summarization methods.

4.3 Statistical and hybrid graph summarization

Several follow-ups on the SNAP summarization approach (Sect. 4.1.2) have been proposed in the literature.

The k-SNAP summarization approach [96,97] is an approximation of the SNAP one. It allows setting the desired number k of summary nodes, so that a whole graph can be visualized at different granularity levels, similarly to roll-up and drill-down OLAP operations. A k-SNAP summary is a graph of k summary nodes which satisfy the above condition (i) of a SNAP summary, but relax condition (ii) so that only some (not every) input graph node represented by some summary node satisfies it. Further, as many such summaries may exist, a k-SNAP summary is defined as one that best satisfy the condition (ii) of SNAP summaries. Finding such a summary is NP-complete [96]; hence, tractable heuristic-based algorithms are proposed to compute approximations of k-SNAP summaries.

The k-SNAP summarization approach has been further extended [114] to handle numerical attributes, while k-SNAP as well as SNAP before have only considered categorial attributes whose domains are made of a limited number of values. The proposed CANAL approach allows bucketizing the values of some numerical attribute into the desired number of categories, hence reducing the summarization of graphs with categorial and numerical attributes to that of graphs with categorial attributes only. For a given numerical attribute a, each of the obtained categories represent a range of a values that nodes with similar edge structure have. Computing such categories is in $O(N \log N + k_a^4)$, where k_a is the number of distinct a values that the input graph contains. Also, to ease the use of k-SNAP to inspect a graph in an roll-up and drill-down OLAP fashion, Zhang et al. [114] provides a solution to automatically recommend k values for visualizing this graph. It consists in ranking k-SNAP summaries of varying k according to a so-called interestingness measure, defined in terms of conciseness, coverage and diversity criteria.

k-SNAP has also strongly inspired the summarization approach in [60], which similarly aims at computing graph summaries w.r.t. user-selected number of summary nodes, attributes and edge types. The summaries are computed using a variant of one above-mentioned tractable k-SNAP heuristics, which keeps the SNAP condition (i) but changes the SNAP condition (ii) that k-SNAP tries to best satisfies, so that a summary best reflects the organization in social com-

[5] https://github.com/yikeliu/VoG-Overlap.

munities of the input graph nodes w.r.t. the selected attributes and edge types.

From a different perspective, [85] sketches SAP HANA's approach for large graph analytics through summarization. It consists in defining rules to summarize part of an analyzed graph. Rules are made of two components, one graph pattern to be matched on the graph, and how the matched data should be grouped and aggregated into a result graph.

5 Structural RDF summarization

Structural summarization of RDF graphs aims at producing a summary graph, typically much smaller than the original graph, such that certain interesting properties of the original graph (connectivity, paths, certain graph patterns, frequent nodes etc.) are preserved in the summary graph. Moreover, these properties are taken into consideration to construct a summary. The methods for structural summarization are distinguished into two categories. The quotient summarization methods, discussed in Sect. 5.1, while the remaining structural summarization methods are described in Sect. 5.2

5.1 Structural quotient RDF summaries

We begin with summarization techniques that are based on quotient methods. Intuitively, each summary node corresponds to (represents) multiple nodes from the input graph, while an edge between two summary nodes represents the relationships between the nodes from the input graph, represented by the two adjacent summary nodes. Often, the nodes formulated in summaries like these, are called *supernodes*, while their edges are called *super-edges*.

An interesting property, which directly follows from the notion of quotient graph, relates query answers on an RDF graph G to query answers on its quotient summary:

Definition 4 (*Representativeness*) Given an *RDF query language (dialect)* $\mathcal{Q}$, an RDF graph G and a summary $\texttt{Sum}$ of it, $\texttt{Sum}$ is *$\mathcal{Q}$-representative of* G if and only if for any query $Q \in \mathcal{Q}$ such that $Q(G^{\infty}) \neq \emptyset$, we have $Q(\texttt{Sum}^{\infty}) \neq \emptyset$.

Informally, representativeness guarantees that queries having answers on G should also have answers on the summary. This is desirable in order for the summary to help users formulate queries: the summary should reflect all graph patterns that occur in the data.

An overview of the structural quotient summaries is shown in Table 3. Section 5.1.1 introduces summaries defined based on bisimulation graph quotients (recall Definition 2), while Sect. 5.1.2 discusses other quotient summaries.

5.1.1 (Bi)simulation RDF summaries

The classical notion of bisimulation (Sect. 4.1) has been used to define many RDF structural quotient summaries.

Thus, [77] presents SAINT-DB, a native RDF management system based on structural indexes. This index is an RDF quotient *simulation*, based on *triple (not node) equivalence*. The summary is not an RDF graph: its nodes group triples from the input, while edge labels indicate positions in which triples in adjacent nodes join. Thus, the index is tailored for reducing the query join effort, by pruning any dangling triples which do not participate in the join. Since the index contains only information on joins, and nothing of the values present in the input graph, the query language is restricted to BGPs comprising of *variables in all positions*; further, these BGPs must be *acyclic*. For compactness, they bound the simulation, for small k values, e.g., 2; this enables compression factors of about 10^4. Semantic information or ontologies are not considered. The time complexity of the algorithm comes from the corresponding algorithms for computing graph simulation. This is $O(N^2 * M)$, where N is the number of nodes (equivalent types) and M is the number of edge labels in the result graph. In practice, different query processing strategies aimed at join pruning are implemented by integrating the structural index with the RDF-3X [70] engine.

A structure-based index is proposed in [98], defined as a bisimulation quotient; the authors show that the summary is representative of only tree-shaped queries over non-type and non-schema triples, comprising a single distinguished variable which corresponds to the root node. Further, the authors study limited versions of the bisimulation quotient by considering: (i) only forward bisimulation, (ii) only backward bisimulation or (iii) only neighborhoods of a certain length for tree structures of the input graph. The proposed applications of the structure index are twofold: (i) for data partitioning, by creating a table for each node of the structure index, thereby physically grouping triples with subjects that share the same structure, and (ii) for query answering, where the query may be run first on the structure index, to obtain the set of candidate answers, thus achieving pruning of the (larger) original graph. The authors do not consider graph semantics, nor answering queries over type and schema triples. The complexity of the corresponding algorithm for generating the index is $O((N_1 \cup N_2) * M * \log N)$, where N_1 and N_2 are the nodes selected for backward and forward bisimulation, respectively, M is the number of edges and N the number of nodes of the input graph.

RDF summaries defined in [17] are quotients based on FW bisimulation. The authors do not consider graph semantics or ontologies. They show how to use the summary as a support for query evaluation: incoming *navigational* SPARQL queries are evaluated on the summary, then the results on the

Table 3 Structural quotient RDF summaries

Work	RDF input component	Input requirements	Purpose	Output type	Output nature	System–theory
ExpLOD [42] [43]	Instance	None	Data exploration, visualization	Graph	Instance and schema	System
Campinas et al. [11]	Instance	None	Query formulation	RDF graph	Instance and schema	Theory
Consens et al. [17,45]	Instance	None	Query answering	RDF graph	Instance and schema	System
Khatchadourian et al. [44]	Instance	None	Data exploration, visualization	Graph	Instance and schema	Theory
Schatzle et al. [86]	Instance	None	Graph reduction	Graph	Instance and schema	Theory
ASSG [113]	Instance	Required user-selected queries	Query answering	Graph	Compressed graph	Theory
Čebirić et al. [12]	Instance and schema	None	Query optimization, query formulation, visualization	RDF graph	Instance and schema	System
Jiang et al. [34]	Instance	None	Semantic mining	Labeled graph	Instance	Theory
Picalausa et al. [77]	Instance	None	Indexing, query answering	Graph	Instance and schema	System
Tran et al. [98]	Instance	Optional parameters FW/BW/FB and neighborhood size	Indexing, data partitioning, query processing	Graph	Instance and schema	Theory

summary are transformed into results on the original graph by exploring the extents of summary nodes. They propose in [45] an implementation based on GraphChi [52], the single-machine multi-core processing framework, to construct the summary in roughly the amount of time required to load the input KB plus write the summary. GraphChi supports the Bulk Synchronous Parallel (BSP) [105] iterative, node-centric processing model, by which nodes in the current iteration execute an update function in parallel, depending on the values from the previous iteration. Their summarization approach is based on the parallel, hash-based approach of [6] which iteratively updates each node's block identifier by computing a hash value from the node's signature defined by the node's neighbors from the previous iteration. The main idea is that two bisimilar nodes will have the same signature, the same hash value, and thus have the same block identifier. Due to the large size of the resulting bisimulation summary, the authors propose a so-called *singleton optimization*, which involves removing summary nodes representing only one node from G; the reduced summary is therefore no longer a quotient of G.

ExpLOD [42,43] produces summaries of RDF graphs, by first transforming the original RDF dataset into an unlabeled-edge-ExpLOD-graph, where a node is created for each triple in the original RDF graph, labeled with the triple property; unlabeled edges go from the original triple's subject and object, to the newly constructed property node. Then, the ExpLOD graph is summarized by a forward bisimulation quotient, grouping together nodes having *the same RDF usage*. RDF usage can be *statistical*, e.g., the number of instances of a particular class, or the number of times a property is used to describe resources in the graph. RDF usage can also be *structural*, e.g., the set of classes to which an instance belongs, the sets of properties describing an instance, or sets of resources connected by the *owl:sameAs* property. As such, they do not propose one summary, but rather a framework where one can select the summary according to his "usage" preferences. Finally, the bisimulation quotient is applied without taking into account neither schema

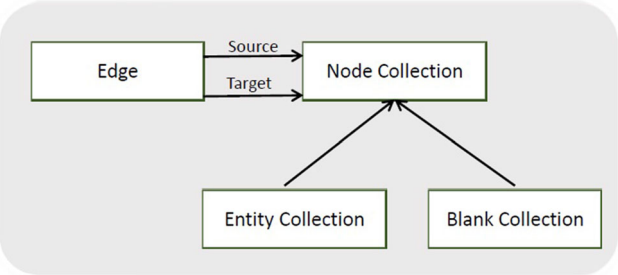

Fig. 6 RDF vocabulary for the data graph summary [11]

nor type triples; thus, the summary is not representative. There are two sequential implementations of ExpLOD. The first implementation computes the relational coarsest partition of a graph using a partition refinement algorithm [71] and requires datasets to fit in main memory. The second approach uses SPARQL queries against an RDF triple store; although in principle this is more scalable, as datasets need not be stored in main memory, it is slower due to the query answering time. To overcome the limitation of the centralized approach, the authors extend ExpLOD, proposing a novel, scalable mechanism to generate usage summaries of billions of Linked Data triples based on a parallel Hadoop implementation [44].

Schätzle et al. [86] considers the problem of efficiently building quotient summaries of RDF graphs based on the FW bisimulation node equivalence relation. The authors do not reserve any special treatment to RDF type and schema triples, which prevents the resulting RDF summaries of being representative. Two implementations of the algorithm for computing graph bisimulations, first introduced in [3], are presented: one for sequential single-machine execution using SQL, and the other for distributed execution, taking advantage of MapReduce parallelization to reduce running time. They both have worst-case time complexity of a $O(M * N + N^2)$.

5.1.2 Other structural quotient summaries

To assist users whose task is query formulation, Campinas et al. [11] creates the summary graph, the so-called *node collection layer*, by grouping nodes having the exact same types, or in the absence of types, the same outgoing properties, into *entity nodes*; further, nodes from the input with no outgoing properties, and having the same incoming properties from subjects with the same set of types, are represented by *blank nodes* (Fig. 6). An edge exists between two summary nodes v_1 and v_2, labeled by a property p, if there exist two nodes v_1' and v_2' in G, such that v_1' is represented by v_1, v_2' is represented by v_2, and there exists an edge labeled by p in G from v_1' to v_2'. The number of represented nodes from the input is attached to each summary node, and the number of represented edges from the input to each summary edge. This summary graph may group resources from multiple datasets. The proposed *dataset layer* groups together nodes of the node collection layer which belong to the same dataset. Schema triples are not considered. The approach bears similarities with ExpLOD, since nodes in the first layer are partitioned by types, and partitions are represented by distinct summary nodes. However, unlike ExpLOD, the G nodes having a type are not further distinguished by their data properties, i.e., two nodes of the same type A, one having the data properties a, b and c and the other having the properties a and d will be represented by the same summary node. Unlike ExpLOD, their summary graph is an RDF graph.

RDF summaries are defined in a rather restricted setting in [34]. The authors assume that all subjects and objects are typed, and that each has exactly one type; class and property URIs are not allowed in subject and object position, and no usage is made of possible schema triples. Under this hypothesis, they construct from the RDF graph a *typed object graph* (TOG) comprising $(\mathtt{s}, \mathtt{p}, \mathtt{o})$ triples and assigning an RDF type for each such $\mathtt{s}$ and $\mathtt{o}$. Two methods are proposed for summarizing the TOG, namely, *equivalent compression* and *dependent compression*. The equivalent compression produces a quotient of the TOG by grouping together nodes having the same type and the same set of labels on the edges adjacent to the node. In the dependent compression, two nodes v_1 and v_2 of the TOG are grouped together if v_1 is adjacent only to v_2, or vice-versa. As application scenarios of this approach the authors indicate mining semantic associations, usually defined as graph or path structures representing group relationships among several instances.

Based on query-preserving graph compression [22] (Sect. 4.1.2), an Adaptive Structural Summary for RDF graphs (ASSG, in short) is introduced in [113]. ASSG aims at building compressed summaries of the part of an RDF graph which is concerned by a certain set of queries. The authors compute a structural rank of nodes, which is 0 for leaves, and grows with the shortest distance between the node and a leaf; then, nodes having the same label and the same rank are considered equivalent, and are all compressed together in a single ASSG node. To partition the N nodes of a graph G to different equivalent classes by their label and rank the cost is $O(N + M)$, where M the number of the edges of the graph.

RDF summaries defined in [12–14] adapt the idea of quotient summaries to two characteristic features of RDF graphs: (i) the presence of type triples (zero, one, or any number of types for a given resource), and (ii) the presence of *schema triples*. As we explained in Sect. 2.1, RDF Schema information is also expressed by means of triples, which are part of G. [12] shows that quotient summarization of *schema* triples does more harm than good, as it destroys the semantics of the original graph. To address this, they introduce a notion of RDF node equivalence which ensures that class and prop-

erty *nodes* (part of schema triples) are not equivalent to any other G nodes, and define a summary as the quotient of G through one such RDF node equivalence. Such summaries are shown to preserve the RDF Schema triples intact, and to enjoy representativeness (Definition 4) for BGP queries having variables in all subject and object positions. The authors show how bisimulation summaries can be cast in this framework, and introduce four novel summaries based *property cliques*, which generalize property co-occurrence as follows. A clique c_G is a set of data properties from G such that for any $p_1, p_2 \in c_G$, a resource of G is the source and/or target of both p_1 and p_2. For instance, if resource r_1 has properties a and b, while r_2 has b and c, then a, b, c are part of the same *source clique*; if, instead, r_1 and r_2 are targets of these properties, then a, b, c are part of the same *target clique*. The so-called weak summary groups together nodes having the same source *or* target clique, while the strong summary requires the same source *and* the same target clique; their variant typed-weak and typed-strong summaries first group nodes according to their types, and then according to their cliques. All these summaries can be computed in linear time in the size of the input graph. A benefit of this specific approach is that clique summaries are orders of magnitude more compact than bisimulation summaries. The authors also study how to obtain the summary of G's semantics, which is the saturation of G. They provide a sufficient condition on the RDF equivalence relation, ensuring that the summary of G^∞ can be computed from G without saturating it, and show that this may be many times faster than the direct procedure of first saturating G, then summarizing G^∞. The software tool implementing this approach is available as open source.[6]

5.2 Structural non-quotient RDF summaries

Several structural RDF summaries have been based on techniques different from structural quotients. Our presentation below attempts to identify families of proposals based on the summarization techniques and/or, as appropriate, on the usage for which the summaries are built. An overview of all structural, non-quotient summaries is shown in Table 4, depicting their individual characteristics as well. Section 5.2.1 presents proposals based on text summarization and information retrieval; Sect. 5.2.2 describes summaries built around concepts of centrality (or rank, importance) of nodes in a graph; Sect. 5.2.3 considers structural RDF summarization based on an index or other structures which aim at selective data access; finally, Sect. 5.2.4 discusses RDF summaries whose goal is to facilitate the extraction of a schema, understood as a compact structural description, of the input RDF graph.

[6] https://gitlab.inria.fr/cedar/quotientSummary.

5.2.1 Inspired by text summarization and information retrieval

One way of summarizing data, especially when the summary is meant for human users, is to select a most significant subset thereof. Such summarization is very useful, considering that the human ability to process information does not change as the available data volumes grow. We describe here RDF knowledge base summarization efforts inspired from information retrieval and text summarization.

In [95], the authors study the problem of selecting the most important part of an RDF graph which is to be shown to a user interested in a certain entity (resource). A fixed space (triple) budget is to be used; beyond labels, authors also allow the edges of the RDF graph to carry weights, with higher-weight edges being more important to show. The authors provide algorithms which select triples favoring closeness to the target entity and weight; then, they extend this with criteria based on diversity (include edges with different labels in the selection) and popularity (favor frequently occurring edge labels). It is worth noting that similar techniques have recently been included in Google's search engine, when the user searches for an entity present in Google's Knowledge Graph, and is presented with a small selection of this entity's properties.

Besides this, text summarization principles, where a text can be seen as a collection of terms or a bag of sentences, have been applied to summarize ontologies. An ontology summarization method along these lines is introduced in [116], based on *RDF Sentence Graphs*. An RDF Sentence Graph is a weighted, directed graph where each vertex represents an RDF sentence, which is a set of RDF Schema statements as illustrated in Fig. 7. A link between two sentences exists, if an object of one sentence belongs to another sentence as well. The creation of a sentence graph is customized by domain experts, who provide as input the desired summary size, and their navigation preferences, i.e., weights in the links they are mostly interested in. Then, the importance of each RDF sentence is assessed by determining its centrality in the sentence graph. The authors compare different centrality measures (based on node degree, betweenness, and the PageRank and HITS scores, and show that weighted in-degree centrality and some eigenvector-based centralities produce better results. Finally, the most important RDF sentences are re-ranked considering the coherence of the summary and the coverage of the original ontology, and the constructed result is returned to the user.

This method does not handle implicit information (thus, it should be applied to a saturated ontology); also, it does not consider the instance graph.

Subsequently, the authors extended their technique from one ontology, to the global set of ontologies harvested from the semantic web [115]. Specifically, to decide the impor-

Table 4 Structural non-quotient RDF summaries

Work	RDF input component	Input requirements	Purpose	Output type	Output nature	System–theory
SchemEX [49,50]	Instance	Data stream window size	Indexing	RDF graph	Instance	Theory
RDF Sentence Graph [115,116]	Schema	Required schema, Parameterized user input, RDF/OWL	Visualization	Labeled graph	Schema	System
KCE [66,75]	Instance and schema	Required schema, Parameterized user input, RDF/OWL	Visualization	Isolated nodes	Schema nodes	System
RDFDigest [74, 99,101]	Instance and schema	Required schema, Parameterized user input, RDF/OWL, Semantics-aware, Handle implicit data	Visualization, query answering tasks	Labeled graph	Schema	System
Queiroz et al. [81]	Schema	Required schema, Parameterized user input, RDF/OWL	Visualization	Labeled graph	Schema	System
Sydow et al. [95]	Instance	Entity of interest	Visualization	RDF graph	Instance	Theory
Gurajada et al. [29]	Instance	None	Query answering	RDF graph	Instance	System
Kellou et al. [39]	Instance and Schema	None	Schema discovery	Graph	Schema	Theory
Le et al. [54]	Instance	Required neighborhood size	Indexing, keyword queries	Partitioned RDF graph	Instance	Theory
Udrea et al. [104]	Instance	Assumes saturated input graph, Required size k of the input graph partition	Indexing, visual querying	Balanced binary tree (leaves = partition over resources)	Instance	System

tance (*salience*) of an RDF sentence, they extend it with *neighboring information*, for example counting how often the terms of the sentence are linked or instantiated in the global semantic web. Two salience measures are proposed: structural and pragmatic importance. *Structural importance* measures the number of entities in the web that have a reference to the local RDF sentences with regard to subjects, predicates or objects. Secondly, the *pragmatics importance* takes into account the statistical frequency of terms instantiated across the global semantic web. The two measures are combined in order to produce an integrated importance value for each RDF sentence, which again is passed to a re-ranking step to ensure coverage over the whole ontology. Although, in the second approach, statistical information over the instances is considered, the approach still does not consider implicit information.

KCE [66,75], on the other hand, attempts to automatically identify the key concepts in an ontology. To achieve this, it combines cognitive principles with lexical and topological measures (the density and the coverage). The goal is to identify a number of concepts that would be selected by human experts. To this direction a number of criteria are defined:

– The notion of *natural category* is used for identifying concepts that are information-rich in a psycho-linguistic sense. This is approximated by two measures: (i) *name*

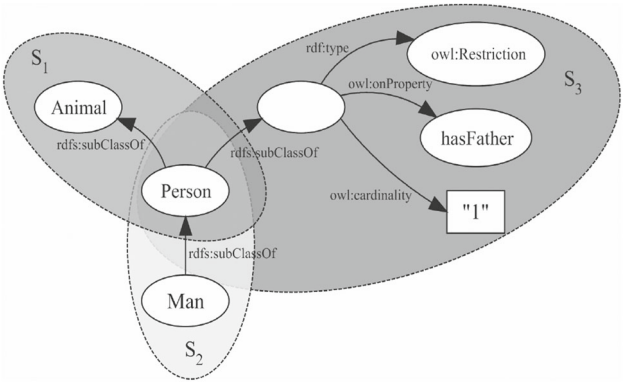

Fig. 7 Sample RDF sentences [116]

simplicity promotes concepts labeled with simple names and penalizes compounds; (ii) *basic level* measures how central a concept is in the taxonomy of the ontology. This is combined with the *density* favouring concepts which have many properties and taxonomic relationships.
- The *coverage* tires to ensure that no part of the ontology is neglected.
- Lastly, the notion of *popularity*, is based on lexical statistics, and tries to identify concepts commonly used in natural language.

Each ontology concept is assigned a score, which is a weighted sum of the scores assigned for each individual criterium; then, the key concepts of the ontologies are taken to be those with the highest score. This approach extracts only schema elements, based on both schema and instance information. Implicit information in the ontology is also taken into account in the process. To fine-tune result quality, the method requires the user to specify values for a set of parameters. The tool implementing this approach is available online and in open source.[7]

5.2.2 Focused on centrality measures

In this section, we present approaches that focus on exploiting centrality measures (and in some cases in combination with other parameters)in order to produce summaries.

RDFDigest [99,101] produces summaries of RDF schemas, consisting of the most representative concepts of the schema, seen also from the angle of their frequency in a given instance (RDF graph). Thus, the input of the process includes both an ontology and a data graph. The tool starts by saturating the knowledge base with all implicit data and schema information, thus taking them into account for the rest of the process. The goal of the work is to identify the most important nodes in the ontology, and to link them in order to produce a valid subgraph of the input schema.

In its first version [100], node *relevance* is defined based on the relative cardinality, and the in/out degree centrality of the node. Then, the most relevant nodes are retained as being part of the answer. To find out how to connect these nodes in a subschema, two algorithms are proposed, trying to maximize the importance of the selected ontology edges either globally or locally. In the first case, a spanning tree is calculated maximizing the importance of the selected edges, and then, the most important nodes are connected using paths from this tree. In the second case, the edges linking the most important nodes are selected based on the notion of coverage trying to maximize the most representative edges out of the whole schema graph. The authors report that linking the most important nodes based on maximum-cost spanning trees produces better summaries according to their experiments with both methods having a worst-case time complexity of $O(N_O^3)$, where N_O is the number of nodes in the ontology. However, this method does not guarantee that the total weight of the selected subgraph is maximized, and when picking a connecting path, it may introduce many additional nodes in the result, some of which may not be important at all. The size and quality of the resulting summary can be fine-tuned by specifying values for a set of parameters; the system is available online[8] and is mostly targeted for visually presenting the ontology summaries.

The more recent version [74] tries to identify the most important schema nodes using six well-known measures from graph theory (i.e., degree, betweenness, bridging centrality, harmonic centrality, radiality, and ego centrality [8]) and adapting them for RDF/S KBs in order to consider instance information as well. The authors model the problem of linking those nodes as a graph Steiner Tree selection problem [30], trying to minimize the total number of additional nodes introduced, employing approximations and heuristics to speed up the execution of the respective algorithms. According to the authors, the optimal selection of importance measure and approximation for linking the most important nodes yields a worst-case time complexity of $O(N^2 * M) + O(S * (N + M))$ where N is the number of nodes, M is the number of edges in the ontology and S the number of most important nodes to be selected. An overview of the summarization process is shown in Fig. 8. Also in this version, users have the opportunity to parameterize the process by specifying values of different parameters, such as the number of summary nodes to be selected. In the latest version of the system [102,103], the authors propose exploration operations based on summaries such as *zoom* and *extend*. Extend focuses on a specific subgraph of the initial

[7] http://www.essepuntato.it/kce.

[8] http://www.ics.forth.gr/isl/rdf-digest/.

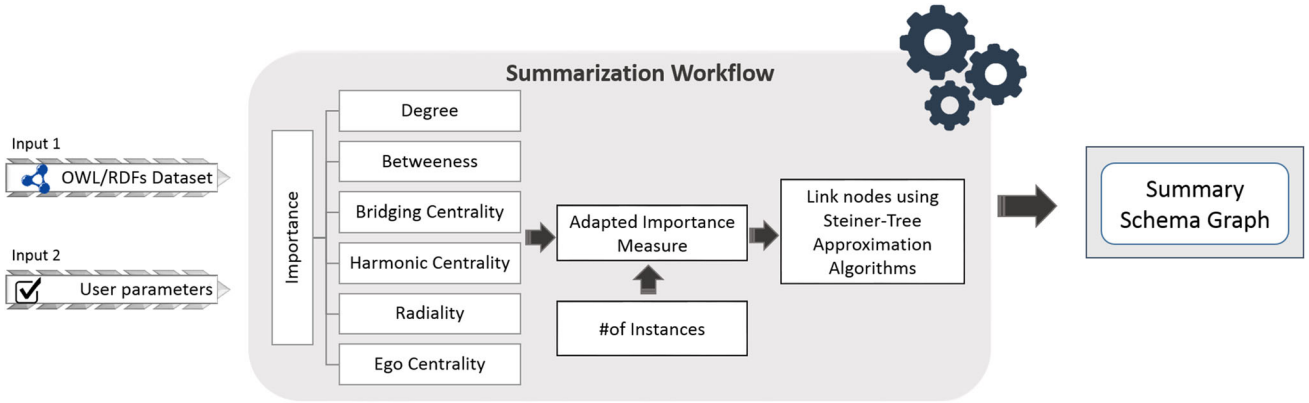

Fig. 8 Creating an ontology schema graph using [74]

ontology, whereas zoom on the whole graph, providing more or less detailed information for the selected nodes.

Queiroz-Sousa et al. [81], on the other hand, tries to combine user preferences with centrality measures in order to calculate the importance of a node. Then, paths that include the most important nodes are identified to produce the final graph. Thus, the result is a subgraph of the original graph. The main steps of this summarization method are shown in Fig. 9: (i) select the parameters (e.g., the size of the summary and importance thresholds) and possibly nodes that are important according to user's opinion; (ii) compute the relevance of the concepts in the ontology as the weighted sum of the degree centrality and the closeness centrality; (iii) identify the paths linking the selected nodes using the Broaden Relevant Paths algorithm. The specific algorithm tries to find paths of greatest quality within the summarized graph by considering the relevance of the included nodes in the path. The approach supports RDF or OWL ontologies and mainly aims to help ontology understanding through visualization.

5.2.3 Index-driven RDF summaries

Besides summaries focusing on centrality measures, other approaches try to exploit summaries for indexing.

GRIN [104], for example, is an index for RDF graphs that has been designed for efficient query answering. The semantics of the indexed RDF graph is taken into account by assuming that the input graph is saturated before being indexed; however, the RDF Schema is not part of the index. A GRIN index is a hierarchical clustering of the resources of an RDF graph, modeled as a balanced binary tree. The set of leaf nodes in the tree, form a partition of the set of triples in the RDF graph; each leaf node represents the resources of the triples it holds. Interior nodes are constructed by finding a *center* triple, and a radius value R. An interior node in the tree implicitly represents the set of all vertices in the graph that are within R units of distance (i.e., less than or equal to R links) from that center. Inner nodes at a same level of the index form a partition of the input RDF graph; each inner node reflects the resources of the triples of the nodes it is an ancestor of. The worst-case complexity for building the index is $O(R^4 * \log_2 R)$. Then, at query time, GRIN derives a set of inequality constraints from the query, evaluated against the nodes of the GRIN index in order to identify the smallest subgraph that contains answers to the input query.

Gurajada et al. [29] studies efficient query processing in the TriAD distributed RDF data management system, by relying on a summary of the RDF graph stored within the system. This summary, which is an RDF graph, follows from a standard partitioning of the input RDF graph, computed with METIS,[9] which minimizes the number of edges across the partition's blocks. The summary is made of triples of the form $B_1 \xrightarrow{p} B_2$ reflecting that there exists a triple (s, p, o) in the input graph with s a resource of the partition block B_1 and o a resource of the block B_2. The proposed technique does not consider implicit triples of the input RDF graph and basically reflects how (some resources of) sets of strongly connected triples, i.e., those reflected by the partition blocks, are loosely connected. Then, at query time, the index is used to prune dangling triples by identifying the bindings for the join variables in the query, which are later used to generate results via the relational joins of the underlying system.

Le et al. [54] builds RDF graph summaries meant to help answer keyword queries in RDF graphs. As common in this setting, a match is a tree of interconnected RDF triples, such that each keyword from the query is present in one of the triples; the score of a query match is consisted of several keywords and decreases with the number of edges (triples) in this tree, that is, the closest the nodes (thus, the smaller the tree) the better; and the answer to the query is the set of the highest-scoring matches. To enable efficient query answering, the authors build a summary as a collection of tree structures (see below), each representing a subset of the graph triples; the summary trees, together, represent all graph

[9] http://glaros.dtc.umn.edu/gkhome/metis/metis/overview.

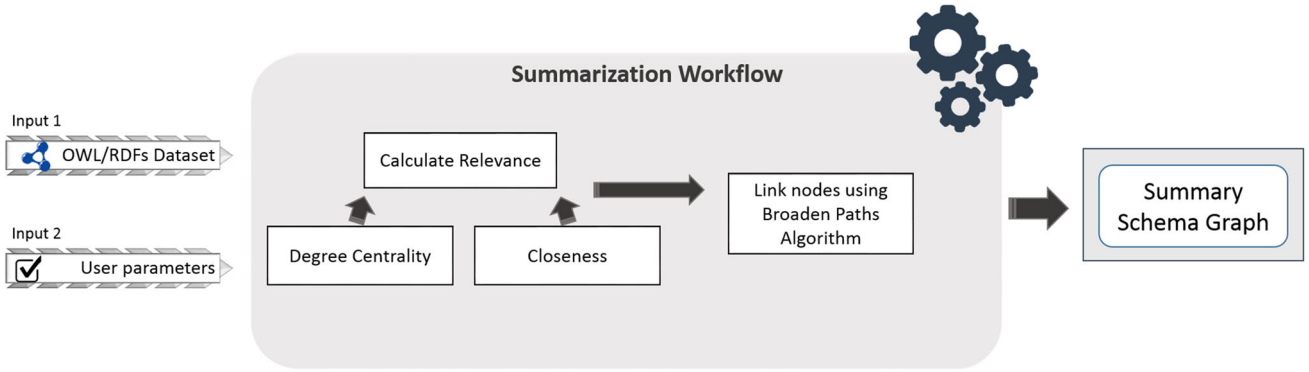

Fig. 9 Flowchart of the ontology summarization method [81]

triples. Given a summary tree, the authors estimate the length of a path which may connect the tree with the nodes matching different query keywords. If the result tree has a length larger than one already found, there is no reason of visiting the nodes of this tree. To summarize an RDF graph (the authors consider a restricted setting where each resource has at most one type), they proceed as follows. (i) For each type T and each resource r of this type, given a user-specified integer parameter α, a partition block is created comprising of the triples forming paths of length at most α from r, except the triples included in previously built partition blocks. (ii) Next, they search for a partition block (subgraph of G) which can be embedded via homomorphism into another, and when this happens, they discard the former (as it can be considered to be "sufficiently well represented" by the latter). More specifically, each partition is represented by its graph core, and homomorphisms are identified between cores; for efficiency, covering trees of subgraphs are used instead of the subgraphs themselves. Overall, this technique does not consider RDF Schema nor implicit information.

5.2.4 Oriented toward schema extraction

In this subsection, we focus on methods that try to create schema-like structures of the available data sources.

One of the challenges when working with Linked Open Data is the lack of a concise schema, or a clear description of the data that can be found in the data source. SchemEX [49,50], an indexing and schema extraction tool for the LOD cloud, attempts to solve this problem. Out of an RDF graph, it produces a three-layered index, based on resource types. Each layer groups input data sources of the LOD cloud into nodes, as follows: (i) in the first layer, each node is a single class c from the input, to which the data sources containing triples whose subject is of type c are associated; (ii) in the second layer, each node, now named as *an RDF type cluster*, is a set of classes C mapped to those data sources having instances whose exact set of types is C; (iii) in the third layer, each node is an equivalence class, where: two nodes u and v from the input belong to the same equivalence class if and only if they have the exact same set of types, they are both subjects of the same data property p, and the objects of that property p belong to the same RDF type cluster. Further, each equivalence class is mapped to all data sources comprising of triples $(\mathtt{s}, \mathtt{p}, \mathtt{o})$ from an input RDF graph, such that $\mathtt{s}$ belongs to the equivalence class of the node. To build the index, a stream-based computation approach is proposed, depicted in Fig. 10. The restriction to a certain window size of the data stream typically leads to incomplete results, thus the choice of the appropriate window size is an essential parameter for the quality of the extracted index. The specific approach does not consider implicit triples, nor untyped resources. In addition, the resulting index is not a quotient, since in each layer data sources may be mapped to multiple index nodes (while a quotient partitions the graph nodes). Finally, the time complexity of the approach is $O(N * \log N)$ or $O(M * \log M)$, where N is the number of RDF classes available and M is the number of properties.

Toward the same goal of building a compact representation of an RDF graph to be used as a schema, [39] proposes an approach to extract a schema-like directed graph, as follows. A density-based clustering algorithm is used on the input RDF graph to identify the summary nodes: each such node, called a (derived) *type*, corresponds to a set of resources with sufficient structural similarity. Further, each of these types is described by a *profile*, i.e., a set of (property, probability) pairs of the form $(\overrightarrow{p}, \alpha)$ or $(\overleftarrow{p}, \alpha)$ meaning that a resource of that type has the outgoing or incoming property p with probability α. These profiles are used to define output schema edges of two kinds: (i) There is a p-labeled edge from a type node T_1 to a type node T_2, i.e., $T_1 \xrightarrow{p} T_2$, whenever p is an outgoing property of T_1's profile and an incoming property of T_2's profile. (ii) There is an rdfs:subClassOf-labeled edge from a type node T_1 to a type node T_2, i.e., $T_1 \xrightarrow{\text{rdfs:subClassOf}} T_2$, whenever T_1 is found more specific than T_2 by an ascending hierarchical clustering algorithm applied to their profiles. The time complexity of the corresponding algorithm is $O(N^2)$ where N is the number of entities in the dataset. The out-

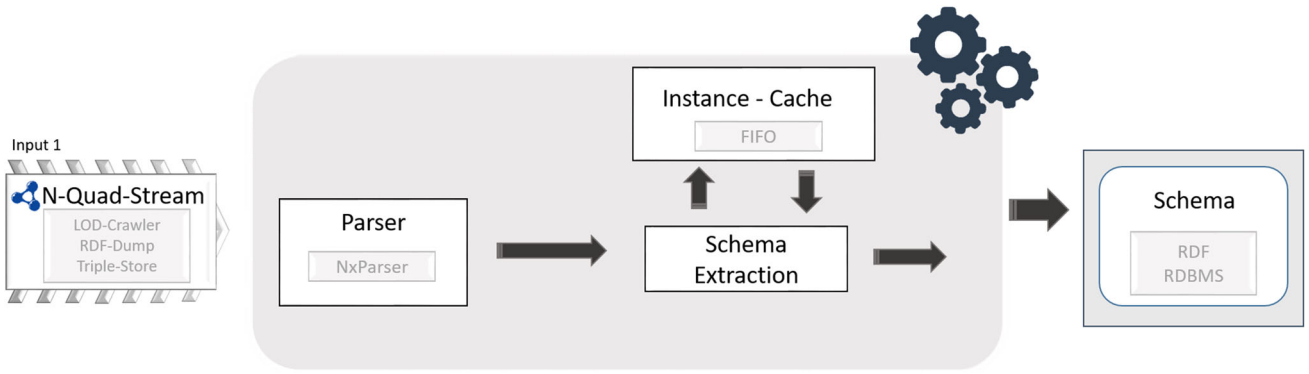

Fig. 10 Graph compression technique for SchemEX

put directed graph can be seen as a summary describing first, types which correspond to structurally similar resources, and second, how properties relate resources of various types.

6 Pattern-based RDF summarization

In this section, we review summarization methods based on data mining techniques, which extract the frequent patterns from the RDF graph, and use these patterns to represent the original RDF graph. A frequent pattern, usually referred to as a *knowledge pattern* in the RDF/OWL KB context (or simply pattern from now on), characterizes a set of instances in an RDF data graph that share a common set of types and a common set of properties. It is usually modeled as a *star* BGP of the form $\{(x, \tau, c_1), \ldots, (x, \tau, c_n), (x, Pr_1, ?b_1), \ldots, (x, Pr_m, ?b_m)\}$ denoting some resource x having types $c_1, \ldots, c_n$ and properties $Pr_1, \ldots, Pr_m$. Given an RDF graph G, a pattern KP identifies all the G resources that match x in the embeddings of KP into G; the number of such embeddings is called the *support* of KP in G. Patterns identified in such a manner become representative nodes (supernodes).

Example 6 (Knowledge pattern) Consider again our sample RDF graph G presented in Fig. 1. The following knowledge pattern $\{(x, \tau, \texttt{Publication}), (x, \texttt{hasTitle}, y), (x, \texttt{hasAuthor}, z)\}$ has no support in G and a support of 1 in G^{∞} (when the embedding matches x to doi_1).

An overview of the works in this category is shown in Table 5. Section 6.1 discusses methods that exploit mining graph patterns, while Sect. 6.2 is concerned with methods which summarize the RDF graph based on rules derived from mining techniques.

6.1 Mining graph patterns

In this section, we describe methods that exploit patterns that eventually appear in the RDF graph and construct the summary based on these patterns.

Zneika et al. [119,120] present an approach for RDF graph summarization based on mining a set of approximate graph patterns that "best" represent the input graph; the summary is an RDF graph itself, which allows to take advantage of SPARQL to evaluate (simplified) queries on the summary instead of the original graph. The approach proceeds in three steps, as shown in Fig. 11. First, the RDF graph is transformed into a binary matrix. In this matrix, the rows represent the subjects and the columns represent the predicates. The semantics of the information is preserved, by capturing the available distinct types and all attributes and properties (capturing property participation both as subject and object for an instance). Second, the matrix created in the previous step is used in a calibrated version of the PaNDa+ [61] algorithm, which allows to retrieve the best approximate RDF graph patterns, based on different cost functions. Each extracted pattern identifies a set of subjects (rows), all having *approximately* the same properties (columns). The patterns are extracted so as to minimize errors and to maximize the coverage (i.e., provide a richer description) of the input data. A pattern thus encompasses a set of concepts (type, property, attribute) of the RDF dataset, holding at the same time information about the number of instances that support this set of concepts. Based on the extracted patterns and the binary matrix, the summary is reconstructed as an RDF graph, enriched with the computed statistic information; this enables SPARQL query evaluation on the summary to return approximate answers to queries asked on the original graph. The time complexity of the approach is $O(M * N + K * M^2 * N)$, where K is the maximum number of patterns to be extracted, M and N are, respectively, the number of distinct properties and subjects/resources in the original KB. The authors note that the algorithm works equally well on homogeneous and heterogeneous RDF graphs.

In [90], the goal is to discover the k patterns which maximize an *informativeness* measure (an informativeness score function is provided as input). The algorithm takes as input an integer distance in d, which will be used to control the neighborhoods in which we will look for similar entities,

Table 5 Pattern-mining RDF summaries

Work	RDF input component	Input requirements	Purpose	Output type	Output nature	System–theory
Zneika et al. [119,120]	Instance	Optional user parameters	Query answering	RDF graph	Instance and schema	Theory
Joshi et al. [35,36]	Instance	None	Compression	Graph and logical rules	Instance	Theory
Pan et al. [73]	Instance	None	Compression	Graph and logical rules	Instance	Theory
Song et al. [90]	Instance	Bounded hop neighbors d, maximum size of patterns K,	Query answering	Graph patterns	Schema	Theory

and a bound k as the maximum number of the desired patterns. The algorithm discovers the k d-summaries/patterns that maximize the informativeness score function.

The authors use d-similarity to capture similarity between entities in terms of their labels and neighborhood information up to the distance d. Compared to other graph patterns like frequent graph patterns, (bi)simulation-based, dual- simulation-based and neighborhood-based summaries, d-similarity offers greater flexibility in matching, while it takes into account the extended neighborhood, something that provides better summaries especially for schema-less knowledge graphs, where similar entities that are not equivalent in a strict pairwise manner. A node v of the original graph G is attributed to the base graph of the d-summary P if and only if there is a node u of P which has the same label as v and for every parent/child u_1 of u in P, there exists a parent/child v_1 of v in G such that edges (u_1, u) and (v_1, v) have the same edge label. Then the d-summaries are used, e.g., to facilitate query answering.

A d-summary P is said to dominate another d-summary P' if and only if $supp(P) \geq supp(P')$; a maximal d-summary P is one that dominates any summary P' that may be obtained from P by adding one more edge. The algorithm starts by discovering all maximal d-summaries by mining and verifying all k-subsets of summaries for the input graph G, then greedily adds a summary pair $(P, P1)$ that brings the greatest increase to the informativeness score of the summary. The time complexity of the approach is $O(S * (b + N) * (b + M) + \frac{K}{2} * S^2)$, where N, M are, respectively, the total number of nodes (subject and objects) and edges (triples) of the original RDF graph, and S is the number of possible d-summaries whose size is bounded by b.

6.2 Mining rules

Methods described here use rule mining techniques in order to extract rules for summarizing the RDF graph. A common limitation of such methods is that, by design, the summary is not an RDF graph, thus it cannot be exploited using the common set of RDF tools (e.g., SPARQL querying, reasoning etc.)

Joshi et al. [35,36] propose compressing the RDF datasets removing triples that can be inferred using logical and inference rules. Thus, graph decompression infers such triples again, to retrieve the original graph.

This approach, which is depicted in Fig. 12, generates, from a given RDF graph G, an *active graph* G_A containing the triples that adhere to certain logical rules, and a *dormant graph* G_D, which contains the set of triples of the original graph which none of the identified rule can infer. This leads to viewing an RDF graph G as being $R(G_A) \cup G_D$, where R represents the set of rules to be applied to the active graph G_A,

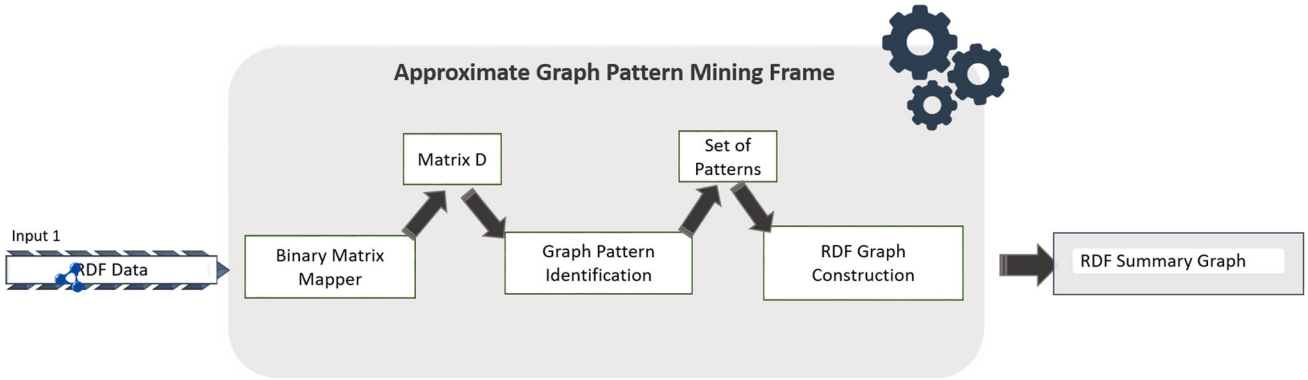

Fig. 11 Zneika et al. [120] approach

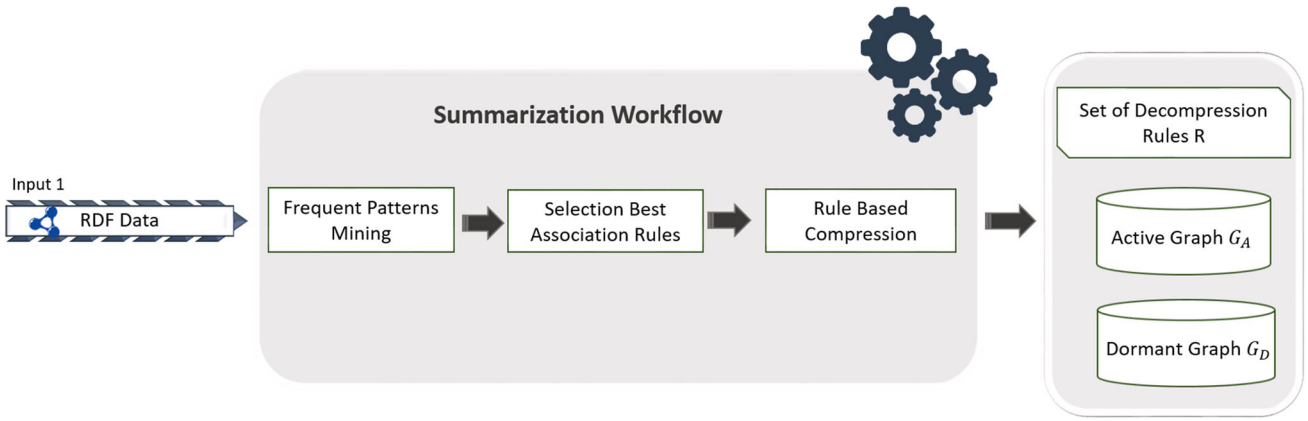

Fig. 12 Graph compression technique for Joshi et al. [36]

while (G_A, G_D) together represent the compressed graph. An association rule mining algorithm is employed to automatically identify the set of logical rules.

The authors leverage the Apriori [2] or FP-Growth [31] frequent pattern-mining algorithms, to identify sets of association rules. First, for each property p, a "transaction" (in classical data mining terms) is a list of objects which are the values of property p for a given subject. Each rule thus is defined by: a property p, an object item k, and a frequent itemset x associated with k. One sample rule is: $\forall x,\ (x, p, k) \rightarrow \bigwedge_{i=1}^{n}(x, p, v_i)$, stating that the subjects that carry the value k for property p, carry also the values u_i for the same property. Based on such a rule, the triple (x, p, k) is encoded in the summary, while the inferred triples $\bigwedge_{i=1}^{n}(x, p, v_i)$ can be removed. Further, the authors extend the approach to use as a transaction, the lists of all (p, o) pairs for a given subject, and similarly mine for frequent itemsets in this context, each of which will be interpreted as a logical compression rule.

The specific approach works well when the original graph contains many different nodes, sharing many same "neighbors", but it is not effective when the contrary is true. To deal with the last issue, the authors of [73] extend the previous approach by exploiting a graph pattern with two variables instead of one, which makes it applicable to more generic graph structures, reducing the size of the summary graph. This is because the number of triples in the summary graph is halved (a rule can now represent more triples). The time complexity of the two previous approaches is $O(M * R + N_p * O_v^2 * N_s)$, where M is the total number of triples, R is the number of the generated logical rules, N_p and N_s are respectively the number of distinct properties and subjects/resources in the graph, and O_v is the average number of different objects/values that are assigned to a property p (we should remember that we are looking for common neighbors and thus for common values for the properties in the object/value part, (s, p, o) or (s, p, v) in the triple notation); thus, the lower the average number of different values the more common the neighborhood and the better the algorithm behaves as already stated earlier (Fig. 13).

7 Statistical summarization

The works we discuss here focus on quantitatively summarization of the contents of an RDF graph. An overview is shown in Table 6.

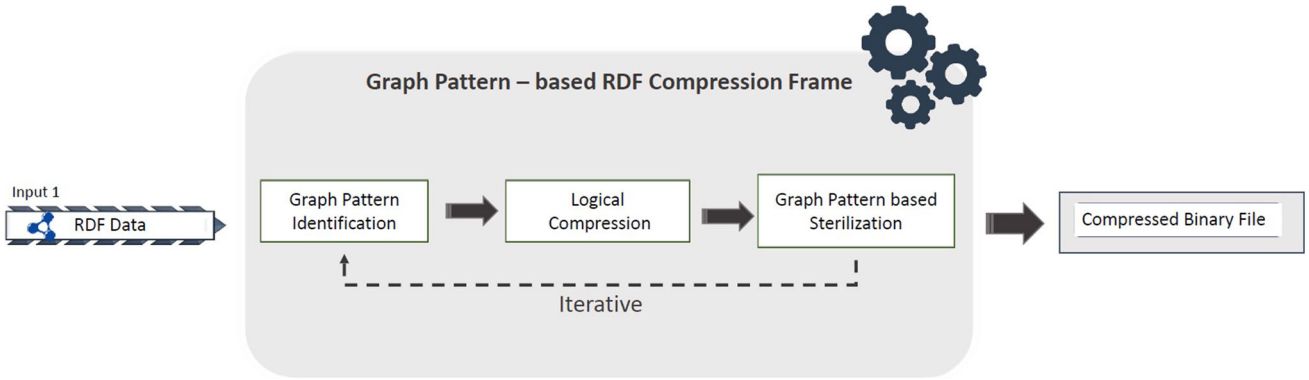

Fig. 13 Graph compression framework following [73]

Table 6 Statistical RDF summaries

Work	RDF input component	Input requirements	Purpose	Output type	Output nature	System–theory
Hose et al. [33]	Instance	None	Source selection	Statistical information	Instance	Theory
Wu et al. [110]	Schema	Requires schema, RDF/OWL, minor user input	Visualization	Isolated schema nodes	Schema	Theory
Pires et al. [78]	Schema	Requires schema, OWL	Query answering tasks	Labeled graph	Schema	System
LODSight [21]	Instance and schema	None	Compression, Visualization	Labeled graph	Instance	System
Mynarz et al. [68]	Instance and schema	Required schema summary patterns, the number (k) of selected examples,	Understanding of dataset, Visualization	Labeled graphs	Instance	System
Presutti et al. [80]	Instance and schema	None	Querying Dataset	Labeled graphs	Construct an ontology and the corresponding instances that summarize key features of dataset and identify the core KPs	Theory

A first motivation for statistical summarization works comes from the source selection problem. In general, statistical methods are interested in providing quantitative statistics on the content of the KBs in order to decide if it is pertinent to use the KB or not. In that respect, compared to the pattern-mining category (which is conceptually close) and the other categories, it has the advantage not to care too much about issues of structural completeness of the summary and thus reducing computational costs. Early works in this area [5,88] use SPARQL ASK queries to identify whether a triple pattern exists in a source node or not, and query those sources only in a subsequent step. The main problem of this solution is that many sources might contain the same facts, meaning that we will have many duplicate results and therefore many unnecessary requests. The authors of [33] propose to expand ASK queries in order not to return a boolean answer, but a concise summary of the result, in the form of Bloom filters [7]. Based on these summaries, a corresponding algorithm estimates the benefit of retrieving results for a triple pattern from a specific source, ignoring sources with low or zero benefit. The summaries produced are called *sketches*, and include statistical

information on the instances. In this approach, input is not required from the users.

Another approach, which seeks to identify the most important resources in an ontology, is [110]. The proposed algorithm, named Concept-And-Relation-Ranking, does not consider instances and tries to identify the most important schema concepts and relations in an iterative manner. The importance of a concept is a combination of the number of relations starting from it, its relations to more important concepts, and the weights of these relations. The more important the concept at the source of a relation, the higher the weight of the relation. The importance of nodes and the weights of the relations reinforce one another in an iterative process. The approach considers implicit and semantic information as well; it is based on an ontology graph model to which RDF, DAML+OIL and OWL ontologies can be mapped.

Pires et al. [78] proposes an method to automatically summarize local ontologies that are used as schemas of peers participating in a peer-to-peer system. The goal is to help peer clustering, where an incoming peer must search for semantically similar peers in order to join. To do that, a schema summary of the new node is compared with the schema summaries of the existing peers in order to decide where to join.

In order to determine the relevance of a concept, two measures are combined: centrality and frequency. Centrality is an adaptation of the degree centrality; different weights can be assigned to user-defined properties, on the one hand, and to the special properties *isA* (RDFS subclassOf), *partOf* and *sameAs*; frequency is the ratio between the number of concept correspondences and the number of distinct local ontologies. The algorithm starts by computing the relevance, then it selects the top-k nodes, and subsequently groups adjacent relevant concepts. Finally, in order to link non-adjacent groups, the first k-paths connecting them are examined in order to select the ones that have the best f-measure and average relevance. The approach does not consider the data triples of the input graph, nor the implicit triples. However, it supports OWL ontologies. Using it requires setting the values of a set of parameters and weights, in particular to determine the importance of different properties, to compute the relevance, to determine the summary size etc. The tool is available online for download.[10]

LODSight [21] is an RDF dataset summary visualization tool that displays typical combinations of types and predicates. It relies solely on SPARQL queries and as such, given a SPARQL endpoint, it can theoretically summarize all accessible data, without requiring any user input. Through those SPARQL queries, it collects statistical information on the available combinations of types and predicates, and visualizes them in a labeled graph. Implicit RDF data are only accounted for to the extent that the endpoint returns full answers based on reasoning. The tool provides dynamic means of changing the level of details and is able to summarize very large datasets. The system is available online.[11]

LODSight was extended in [68] in order to further improve the understanding of a dataset, by instantiating the summary patterns identified by LODSight. To do that, the authors propose an approach to select instances through three methods, namely random, distinct and representative. In random selection, random examples of each RDF summary path are selected; this runs the risk of returning duplicates. The distance selection method aims to select data paths as distinct from one another as possible; to this effect, distance measures are used to find how similar two paths are, and a greedy heuristic is employed to construct a sufficiently diverse set of pairs. The representation selection method combines diversity and representativeness criteria in order to select a set of paths achieving a comprehensive result. The selection method is available as a web service.[12]

Presutti et al. [80] proposes a dataset analysis method based on recognizing and discovering patterns. The aim of this method is to support query answering. As such, the authors create initially an ontology that depicts the organization of the dataset and identifies its main features, i.e., information about triples, paths, and types and properties occurring in the paths. In addition, it includes statistics about these elements, such as the number of occurrences of each path. Using this ontology, the core types and properties can be distinguished based on their frequencies and the position in paths. According to these observations, central knowledge patterns (containing a central type and properties) are extracted in order to define prototypical queries. Implicit information is not taken into account.

8 Other summarization methods

In this section, we present approaches that combine methods from the structural, statistical and pattern-mining categories in order to get better results. In addition, there are methods going beyond RDF graph summaries, for example summarizing DL ontologies. An overview of the works in this category is shown in Table 7.

Alzogbi and Lausen [3] proposes a hybrid structural summarization technique for RDF graphs, the purpose of which is to reduce their size while retaining their structure as much as possible. It consists of a graph quotient step followed by a graph clustering step. The first one adopts *bounded* forward bisimulation, as according to the authors, previous studies showed that, in general, *unbounded* bisimulation is not amenable to significant graph size reduction. This

[10] http://www.cin.ufpe.br/~speed/OWLSummarizer/.

[11] http://lod2-dev.vse.cz/lodsight/about.html.

[12] https://github.com/jindrichmynarz/rdf-path-examples.

Table 7 Hybrid RDF summaries

Work	Method	RDF input component	Input requirements	Purpose	Output type	Output nature	System-theory
Alzogbi and Lausen [3]	Structural quotient, clustering	Instance	None	Compression	Graph	Instance and schema	Theory
Stefanoni et al. [93]	Structural non-quotient, data mining	Instance	Optional parameters for summary refinement	Conjunctive query cardinality estimation	Graph	Instance	Theory
ABSTAT [72] [91]	Statistical, pattern mining	Instance and schema	RDF/OWL, Semantic-aware, Handles implicit data	Visualization, schema discovery	Labeled graph (patterns)	Schema graph (Abstract Knowledge Patterns)	System
Pham et al. [76]	Structural non-quotient, statistical	Instance and schema	Required max. number of summary nodes, min. number of nodes represented by a summary node, infrequency threshold, similarity threshold	Query optimization	Relational tables and foreign keys	Instance	Theory
Zheng et al. [118]	Structural quotient, pattern-mining	Instance and schema	Each instance should have a type, the list of meaning-equivalent instances should be provided	Query optimization	Multi-layer Graph	Schema	Theory
Glimm et al. [25]	Pattern mining	Instance and schema	Description Logics, Semantic-aware, Handles implicit data	Compression	ABox (facts)	Instance	Theory
Fokoue et al. [19,20,23,24]	Structural non-quotient	Instance and schema	Description Logics, Semantic-aware, Handles implicit data	Consistency checking and query answering	ABox (facts)	Instance	System

intermediate graph being a quotient, it represents all the N nodes and M edges of the input graph, and is computed in $O(M*N+N^2)$. It is then further compressed by hierarchical clustering, which fuses root nodes of similar depth-bounded tree subgraphs. This is achieved with the so-called Complete-Link Clustering algorithm in $O(N^2)$ [67], where N is the number of nodes of the aforementioned intermediate graph. This technique produces a standard graph out of an RDF one, without user input, which summarizes the whole input graph (nodes and edges). However, it does not perform particular treatment on RDF Schema triples, hence does not capture the implicit triples they entail, unless input RDF graph are saturated prior to summarization.

ABSTAT [72,91,92] is a summarization method for RDF graphs (and OWL KBs), which aims at reflecting how class instances are related through properties. A summary is not a graph but a set of abstract knowledge patterns (AKPs) of the form (c_1, p, c_2) representing the (s, p, o) graph triples with c_1 (resp. c_2) one of the most specific types of s (resp. o); there may have several such c_1, c_2 pairs for a given property p. An ABSTAT summary is built in polynomial time in the size of the input RDF graph, by first computing for every value presents in the graph all its types, from which are pruned out the redundant ones. Then, for each property assertion (s, p, o), an AKP (c_1, p, c_2) is built if c_1 (resp. c_2) is a most specific type for value s (resp. o). ABSTAT is available online.[13]

Stefanoni et al. [93] proposes to use structural summaries of RDF graphs for estimating the cardinality of conjunctive queries. The authors build a graph summary of an RDF graph by grouping together nodes having exactly the same set of types, same outgoing and same incoming properties. A summary edge is labeled with the number of edges of G that have been collapsed due to merging (thus, the summary is not a quotient). The classes (appearing in the object position of type triples), and properties appearing in the property position are preserved, i.e., they are represented in the summary by themselves. However, properties appearing in the subject/object positions are not preserved; moreover, possible RDFS properties are summarized just as any other data property; thus, the schema is not conserved either. These summaries, built in time linear in M, the number of graph edges, may be too large; therefore, the authors propose an algorithm to reduce the summary to a target size specified by the user, by merging nodes having similar incoming and outgoing properties. The similarity is determined by a Jaccard index, approximated by MinHashing [56]; to efficiently compute the similarity between all pairs of summary nodes, locality-sensitive hashing [56] is used. The approach gets as input only instances and optional parameters for summary refinement and returns an instance graph. This graph is further used to enable the estimation of the cardinality for easing query answering and evaluation.

Pham et al. [76] combines structural non-quotient and statistical methods to create a summary of an RDF graph, which they call *relational schema*. The initial summary is generated, in linear time of the RDF graph size (average complexity), by computing sets of properties joining on the subject, the so-called *characteristic sets*, denoted CS. A summary node is created for each CS, thus representing nodes of G having the same outgoing properties. An edge exists between two summary nodes, labeled by a property p, if there exist two G nodes in their respective extents, such that there exists an edge labeled by p in G between the two G nodes. This structural aspect therefore considers only the instance component of an RDF graph. In the second step, a short human-friendly *label* is computed for summary nodes and edges, by relying on type triples, or in their absence, on ontology information. To reduce the summary size, summary nodes are subsequently merged. In *semantic merging*, two summary nodes can be merged in two ways: (i) if they have the same label, taken from an ontology, or (ii) if their labels, originating from different ontologies, have a common superclass, and the *generality score* of this superclass is lower than a certain threshold. The generality score of an ontology class v is computed as the ratio between the number of instances of v's subclasses and the total number of instances covered by the ontology to which v belongs. Moreover, the two ways in which two summary nodes can be merged in *structural merging* are as follows: (i) if they both have an incoming edge with the same property, from another summary node, or (ii) if the properties in their respective CSs have the TF/IDF similarity score higher than a given threshold. The merging order of the nodes, affects the resulting summary. The summary is modeled relationally: a table is created per summary node, with a column for each property in the CS represented by the summary node; the relationships between the nodes are stored as foreign keys. The chosen relational model proves challenging for storing the possibly highly heterogeneous graph structure, inherent to RDF graphs, and it drives the modifications to the summary. First, the CS of each summary node may be a result of merging multiple summary nodes and thus their CSs. Therefore, a G node in the extent of a summary node may not have a value for each property in the CS of the summary node, and will have a `NULL` value in the table for each such property. Properties having few non-`NULL` values are deleted. Second, a single property in RDF may have multiple literal values, possibly of different types. In such cases, Pham et al. [76] chooses to add a column for each distinct literal value type per property, which other than incurring space costs, may lead to more `NULL` values. Therefore, for each literal value type below the infrequency threshold, all triples are moved to a separate single `PSO` table. Finally, infrequent edges are deleted from the summary, while a summary node

[13] http://abstat.disco.unimib.it/search.

is deleted if the number of nodes it represents is below a threshold; with the exception of summary nodes which are referred to many times from other tables. The approach relies on end-users for choosing the right parameters, whereas the authors propose also an auto-tuning algorithm for determining the best value of the similarity threshold.

Zheng et al. [118] proposes a framework for mining equivalent structure patterns with equivalent semantic meaning. As in RDF KBs, it is common to have different graph structures, sharing the same meaning, the authors' aim is to ease end-user's querying task. As such, instead of demanding from the users to have the complete knowledge of the schema—enumerating in the query all possible semantically equivalent graph structures—the authors propose an approach that performs query rewriting, exploiting automatically other possible graph structures with the same meaning. To achieve that, they define the notion of *semantic graph edit distance* and present a framework that tries first to rewrite the input query to one considering semantic equivalences and then finding the subgraphs minimizing the semantic graph edit distance. For efficiency, they build offline a *semantic summary graph* over which they perform a two-level pruning at query time in order to finally provide answers. The semantic summary graph is a multi-layer graph where the first layer is consisted of the linked types of the instances (they call them semantic facts). Then, they abstract this graph in the layers above, replacing/abstracting in each layer classes with their superclass. The summary produced does not require user input to be produced, whereas the aforementioned method can only be applied in fully typed RDF KBs.

Finally, the next works consider structural methods for the summarization of ABoxes (facts) in description logics KBs.

Glimm et al. [25] proposes a method for compressing (hence summarizing) the ABox of a Horn-$\mathcal{ALCHOI}$ description logic KBs, using the notion of ABox abstraction. Given an ABox $\mathcal{A}$, for each $\mathcal{A}$ value v, a type pattern of the form $tp(v) = (tp_{\downarrow}, tp_{\rightarrow}, tp_{\leftarrow})$ is computed, where $tp_{\downarrow}$ denotes the explicit types v has in $\mathcal{A}$ (C's such that $C(v) \in \mathcal{A}$), $tp_{\rightarrow}$ the outgoing properties v has in $\mathcal{A}$ (R's such that $R(v, v') \in \mathcal{A}$) and $tp_{\leftarrow}$ the incoming properties v has in $\mathcal{A}$ (S's such that $S(v', v) \in \mathcal{A}$). These type patterns are then used to build the abstraction $\mathcal{B}$ of the ABox, which is an ABox itself; each such type pattern is used to represent all the ABox values that match it: for every type pattern $tp = (\{C_1, \ldots, C_m\}, \{R_1, \ldots, R_n\}, \{S_1, \ldots, S_l\})$, $\mathcal{B}$ contains $\{C_1(x_{tp}), \ldots, C_m(x_{tp}), R_1(x_{tp}, y_{tp}^{R_1}), \ldots, R_n(x_{tp}, y_{tp}^{R_n}), S_1(y_{tp}^{S_1}, x_{tp}), \ldots, S_l(y_{tp}^{S_l}, x_{tp})\}$. We remark that, given an ABox, its abstraction is simply obtained by traversing its facts. It is further shown in [25] how the abstraction of the ABox of an input Horn-$\mathcal{ALCHOI}$ KB can be gradually refined, using reasoning steps, to obtain the abstraction of the ABox of the input KB saturation.

ABox summaries has also been considered for the data management tasks of consistency checking [23,24] and query answering [19,20] in $\mathcal{SHIN}$ KBs. In these works, the notion of a summary ABox is very general: an ABox $\mathcal{A}'$ is a summary of another ABox $\mathcal{A}$ w.r.t. some function f that maps $\mathcal{A}$ values to $\mathcal{A}'$ ones whenever f defines a homomorphism from $\mathcal{A}$ to $\mathcal{A}'$. Based on this property of ABox summaries, the purpose of [19,20,23,24] is to study how $\mathcal{SHIN}$ consistency and query answering reasoning techniques can be performed correctly and more efficiently than when handling the input ABox.

9 Conclusions and future work

In this survey, we present a comprehensive state-of-the-art in semantic graph summarization. To this direction, we introduce a taxonomy of the works in the area (based on different properties/criteria that the works adhere to), that can help practitioners and researchers to determine the method most suitable for their data and goal. In this taxonomy, we grouped the main methods of the algorithms presented into four main categories structural, statistical, pattern-mining and hybrid, identifying subcategories whenever possible. In addition, we also classified works in the field according to their input, output, availability on the internet and purpose, showing the rapidly evolving dynamics in the area.

In general, RDF graph summaries can be useful in various application scenarios ranging from data understanding to query answering and from RDF graph data indexing to RDF graph visualization. Structural quotient summaries are most applicable to indexing and query answering through graph reduction; this holds especially for quotients built through equivalence relations such as bisimilarity (possibly bounded). Non-quotient summaries mostly target visualization, schema discovery and data understanding. Pattern-mining summaries provide in many cases logical rules besides the summary graph as part of the final result, so could be possibly more useful in RDF graph compression scenarios. Summaries could also be really useful in data integration scenarios [48], where instead of generating mappings [63,65] between data source schemas, summaries could be used to drive the definition of the mapping. Extending this to a scenario where the sources can also evolve [46,47], summaries can play a key role in schema understanding and mapping redefinition. Different RDF summarization scenarios each bring their very specific requirements (e.g., whether the summary size is bounded or not, whether a schema is present or not, whether to summarize the data or the schema, whether the summary needs to reflect all the structure or not etc.); in many cases more than one algorithm or family of algorithms will provide suitable results. The goal of our survey was to provide enough information to the users of these algorithms

(i.e., application developers or researchers) in order to be able to easily refer to the characteristics of each approach, and evaluate their suitability to their application requirements.

Despite the considerable amount of work in the area of semantic graph summarization, there still are many important open problems in the field. Below we mention two of the most notable ones.

The first one is the quality of the produced RDF summary. Since the result summary for the different algorithms, varies among a selection of nodes, quotients, some other frequency structure or a complete graph with various types of nodes and links identifying a single golden standard is a complex task. On the other hand, even domain experts in many cases disagree on which specific elements should be selected in a semantic summary. However, having in mind our proposed taxonomy, we believe that the next step in the area is the establishment of different golden standards specific to each subcategory focusing on specific purpose input and output.

Another open issue we perceive as really important, is the dynamic nature of all these datasets. As new information becomes available due to new experimental evidence or observations and erroneous past conceptualizations are constantly updated many datasets are rapidly changing. However, summarization in most cases and especially for big data sources is a time-consuming process that should be constantly updated to facilitate data exploration. As such, novel ideas should focus in this dynamicity, augmenting the exploration experience of end-users.

The work in RDF graph summarization gains more importance as the RDF Knowledge Bases become larger and more connected and thus we expect to see additional advances in the field in the near future.

Acknowledgements This research is implemented through IKY scholarships programme and co-financed by the European Union and Greek national funds through the action entitled "Reinforcement of Postdoctoral Researchers", in the framework of the Operational Programme "Human Resources Development Program, Education and Life-long Learning" of the National Strategic Reference Framework (NSRF) 2014–2020.

References

1. Abiteboul, S., Hull, R., Vianu, V.: Foundations of Databases. Addison-Wesley, Reading (1995)
2. Agrawal, R., Imielinski, T., Swami, A.N.: Mining association rules between sets of items in large databases. In: Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, Washington, DC, USA, May 26–28, 1993, pp. 207–216 (1993)
3. Alzogbi, A., Lausen, G.: Similar structures inside RDF-graphs. In: Proceedings of the WWW2013 Workshop on Linked Data on the Web, Rio de Janeiro, Brazil, May 14, 2013 (2013)
4. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F. (eds.): The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press, Cambridge (2003)
5. Basca, C., Bernstein, A.: Avalanche: putting the spirit of the web back into semantic web querying. In: Proceedings of the ISWC 2010 Posters & Demonstrations Track: Collected Abstracts, Shanghai, China, November 9, 2010 (2010)
6. Blom, S., Orzan, S.: A distributed algorithm for strong bisimulation reduction of state spaces. STTT **7**(1), 74–86 (2005)
7. Bloom, B.H.: Space/time trade-offs in hash coding with allowable errors. Commun. ACM **13**(7), 422–426 (1970)
8. Boldi, P., Vigna, S.: Axioms for centrality. Internet Math. **10**(3–4), 222–262 (2014)
9. Bursztyn, D., Goasdoué, F., Manolescu, I.: Efficient query answering in DL-Lite through FOL reformulation (extended abstract). In: Proceedings of the 28th International Workshop on Description Logics, Athens, Greece, June 7–10, 2015 (2015)
10. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Tractable reasoning and efficient query answering in description logics: the DL-Lite family. J. Autom. Reason. **39**(3), 385–429 (2007)
11. Campinas, S., Perry, T., Ceccarelli, D., Delbru, R., Tummarello, G.: Introducing RDF graph summary with application to assisted SPARQL formulation. In: 23rd International Workshop on Database and Expert Systems Applications, DEXA 2012, Vienna, Austria, September 3–7, 2012, pp. 261–266 (2012)
12. Čebirić, Š., Goasdoué, F., Guzewicz, P., Manolescu, I.: Compact Summaries of Rich Heterogeneous Graphs. Research report RR-8920, INRIA Saclay; Université Rennes 1 (2017). https://hal.inria.fr/hal-01325900
13. Čebirić, Š., Goasdoué, F., Manolescu, I.: Query-oriented summarization of RDF graphs. PVLDB **8**(12), 2012–2015 (2015)
14. Cebiric, S., Goasdoué, F., Manolescu, I.: Query-oriented summarization of RDF graphs. In: Proceedings of the Data Science—30th British International Conference on Databases, BICOD 2015, Edinburgh, UK, July 6–8, 2015, pp. 87–91 (2015)
15. Chen, C., Lin, C.X., Fredrikson, M., Christodorescu, M., Yan, X., Han, J.: Mining graph patterns efficiently via randomized summaries. PVLDB **2**(1), 742–753 (2009)
16. Chen, C., Yan, X., Zhu, F., Han, J., Yu, P.S.: Graph OLAP: towards online analytical processing on graphs. In: Proceedings of the 8th IEEE International Conference on Data Mining (ICDM 2008), December 15–19, 2008, Pisa, Italy (2008)
17. Consens, M.P., Fionda, V., Khatchadourian, S., Pirrò, G.: S+EPPs: construct and explore bisimulation summaries, plus optimize navigational queries; all on existing SPARQL systems. PVLDB **8**(12), 2028–2031 (2015)
18. Consens, M.P., Miller, R.J., Rizzolo, F., Vaisman, A.A.: Exploring XML web collections with DescribeX. TWEB **4**(3), 11:1–11:46 (2010)
19. Dolby, J., Fokoue, A., Kalyanpur, A., Kershenbaum, A., Schonberg, E., Srinivas, K., Ma, L.: Scalable semantic retrieval through summarization and refinement. In: Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence, July 22–26, 2007, Vancouver, British Columbia, Canada, pp. 299–304 (2007)
20. Dolby, J., Fokoue, A., Kalyanpur, A., Schonberg, E., Srinivas, K.: Scalable highly expressive reasoner (SHER). J. Web Semant. **7**(4), 357–361 (2009)
21. Dudás, M., Svátek, V., Mynarz, J.: Dataset summary visualization with LODSight. In: The Semantic Web: ESWC 2015 Satellite Events—ESWC 2015 Satellite Events Portorož, Slovenia, May 31–June 4, 2015, Revised Selected Papers, pp. 36–40 (2015)
22. Fan, W., Li, J., Wang, X., Wu, Y.: Query preserving graph compression. In: Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2012, Scottsdale, AZ, USA, May 20–24, 2012, pp. 157–168 (2012)

23. Fokoue, A., Kershenbaum, A., Ma, L.: SHIN ABox reduction. In: Proceedings of the 2006 International Workshop on Description Logics (DL2006), Windermere, Lake District, UK, May 30–June 1, 2006 (2006)
24. Fokoue, A., Kershenbaum, A., Ma, L., Schonberg, E., Srinivas, K.: The summary Abox: cutting ontologies down to size. In: Proceedings of the Semantic Web—ISWC 2006, 5th International Semantic Web Conference, ISWC 2006, Athens, GA, USA, November 5–9, 2006, pp. 343–356 (2006)
25. Glimm, B., Kazakov, Y., Liebig, T., Tran, T., Vialard, V.: Abstraction refinement for ontology materialization. In: Proceedings of the Semantic Web—ISWC 2014—13th International Semantic Web Conference, Riva del Garda, Italy, October 19–23, 2014, Part II, pp. 180–195 (2014)
26. Goasdoué, F., Karanasos, K., Leblay, J., Manolescu, I.: View selection in semantic web databases. PVLDB **5**(2), 97–108 (2011)
27. Goasdoué, F., Manolescu, I., Roatis, A.: Efficient query answering against dynamic RDF databases. In: Joint 2013 EDBT/ICDT Conferences, EDBT '13 Proceedings, Genoa, Italy, March 18–22, 2013, pp. 299–310 (2013)
28. Goldman, R., Widom, J.: Dataguides: enabling query formulation and optimization in semistructured databases. In: VLDB'97, Proceedings of 23rd International Conference on Very Large Data Bases, August 25–29, 1997, Athens, Greece, pp. 436–445 (1997)
29. Gurajada, S., Seufert, S., Miliaraki, I., Theobald, M.: Using graph summarization for join-ahead pruning in a distributed RDF engine. In: Proceedings of the Sixth Workshop on Semantic Web Information Management, SWIM 2014, Snowbird, UT, USA, June 22–27, 2014 (2014)
30. Hakimi, S.L.: Steiner's problem in graphs and its implications. Networks **1**(2), 113–133 (1971)
31. Han, J., Pei, J., Yin, Y., Mao, R.: Mining frequent patterns without candidate generation: a frequent-pattern tree approach. Data Min. Knowl. Discov. **8**(1), 53–87 (2004)
32. Henzinger, M.R., Henzinger, T.A., Kopke, P.W.: Computing simulations on finite and infinite graphs. In: FOCS (1995)
33. Hose, K., Schenkel, R.: Towards benefit-based RDF source selection for SPARQL queries. In: Proceedings of the 4th International Workshop on Semantic Web Information Management, SWIM 2012, Scottsdale, AZ, USA, May 20, 2012, p. 2 (2012)
34. Jiang, X., Zhang, X., Gao, F., Pu, C., Wang, P.: Graph compression strategies for instance-focused semantic mining. In: Linked Data and Knowledge Graph—7th Chinese Semantic Web Symposium and 2nd Chinese Web Science Conference, CSWS 2013, Shanghai, China, August 12–16, 2013. Revised Selected Papers, pp. 50–61 (2013)
35. Joshi, A.K., Hitzler, P., Dong, G.: Towards logical linked data compression. In: Proceedings of the Joint Workshop on Large and Heterogeneous Data and Quantitative Formalization in the Semantic Web, LHD+ SemQuant2012, at the 11th International Semantic Web Conference, ISWC2012. Citeseer (2012)
36. Joshi, A.K., Hitzler, P., Dong, G.: Logical linked data compression. In: The Semantic Web: Semantics and Big Data, 10th International Conference, ESWC 2013, Montpellier, France, May 26–30, 2013. Proceedings, pp. 170–184 (2013)
37. Kaushik, R., Bohannon, P., Naughton, J.F., Korth, H.F.: Covering indexes for branching path queries. In: Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data, Madison, Wisconsin, June 3–6, 2002, pp. 133–144 (2002)
38. Kaushik, R., Shenoy, P., Bohannon, P., Gudes, E.: Exploiting local similarity for indexing paths in graph-structured data. In: Proceedings of the 18th International Conference on Data Engineering, San Jose, CA, USA, February 26–March 1, 2002, pp. 129–140 (2002)
39. Kellou-Menouer, K., Kedad, Z.: Schema discovery in RDF data sources. In: Conceptual Modeling—Proceedings of the 34th International Conference, ER 2015, Stockholm, Sweden, October 19–22, 2015 (2015)
40. Khan, A., Bhowmick, S.S., Bonchi, F.: Summarizing static and dynamic big graphs. PVLDB **10**(12), 1981–1984 (2017)
41. Khan, K., Nawaz, W., Lee, Y.: Set-based approximate approach for lossless graph summarization. Computing **97**(12), 1185–1207 (2015)
42. Khatchadourian, S., Consens, M.P.: Explod: summary-based exploration of interlinking and RDF usage in the linked open data cloud. In: The Semantic Web: Research and Applications, 7th Extended Semantic Web Conference, ESWC 2010, Heraklion, Crete, Greece, May 30–June 3, 2010, Proceedings, Part II, pp. 272–287 (2010)
43. Khatchadourian, S., Consens, M.P.: Exploring RDF usage and interlinking in the Linked Open Data Cloud using ExpLOD. In: WWW2010 Workshop on Linked Data on the Web (LDOW) (2010)
44. Khatchadourian, S., Consens, M.P.: Understanding billions of triples with usage summaries. In: Semantic Web Challenge (2011)
45. Khatchadourian, S., Consens, M.P.: Constructing bisimulation summaries on a multi-core graph processing framework. In: Proceedings of the Third International Workshop on Graph Data Management Experiences and Systems, GRADES 2015, Melbourne, VIC, Australia, May 31–June 4, 2015 (2015)
46. Kondylakis, H., Plexousakis, D.: Ontology evolution in data integration: Query rewriting to the rescue. In: Conceptual Modeling–ER 2011, 30th International Conference, ER2011, Brussels, Belgium, October 31–November 3, 2011. Proceedings, pp. 393–401 (2011)
47. Kondylakis, H., Plexousakis, D.: Ontology evolution: assisting query migration. In: Conceptual Modeling—31st International Conference ER 2012, Florence, Italy, October 15–18, 2012. Proceedings, pp. 331–344 (2012)
48. Kondylakis, H., Plexousakis, D.: Ontology evolution without tears. J. Web Semant. **19**, 42–58 (2013)
49. Konrath, M., Gottron, T., Scherp, A.: Schemex–web-scale indexed schema extraction of Linked Open Data. In: Semantic Web Challenge, Submission to the Billion Triple Track, pp. 52–58 (2011)
50. Konrath, M., Gottron, T., Staab, S., Scherp, A.: Schemex—efficient construction of a data catalogue by stream-based indexing of linked data. J. Web Semant. **16**, 52–58 (2012)
51. Koutra, D., Kang, U., Vreeken, J., Faloutsos, C.: Summarizing and understanding large graphs. Stat. Anal. Data Min. **8**(3), 183–202 (2015)
52. Kyrola, A., Blelloch, G.E., Guestrin, C.: Graphchi: Large-scale graph computation on just a PC. In: 10th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2012, Hollywood, CA, USA, October 8–10, 2012, pp. 31–46 (2012)
53. Lanti, D., Rezk, M., Xiao, G., Calvanese, D.: The NPD benchmark: Reality check for OBDA systems. In: Proceedings of the 18th International Conference on Extending Database Technology, EDBT 2015, Brussels, Belgium, March 23–27, 2015, pp. 617–628 (2015)
54. Le, W., Li, F., Kementsietsidis, A., Duan, S.: Scalable keyword search on large RDF data. IEEE TKDE **26**(11), 2774–2788 (2014)
55. LeFevre, K., Terzi, E.: Grass: Graph structure summarization. In: Proceedings of the SIAM International Conference on Data Mining, SDM 2010, April 29–May 1, 2010, Columbus, Ohio, USA, pp. 454–465 (2010)
56. Leskovec, J., Rajaraman, A., Ullman, J.D.: Mining of Massive Datasets, 2nd edn. Cambridge University Press, Cambridge (2014)
57. Lin, S.D., Yeh, M.Y., Li, C.T.: Sampling and summarization for social networks. In: 17th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD) (tutorial) (2013)

58. Liu, X., Tian, Y., He, Q., Lee, W., McPherson, J.: Distributed graph summarization. In: Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management, CIKM 2014, Shanghai, China, November 3–7, 2014, pp. 799–808 (2014)
59. Liu, Y., Safavi, T., Dighe, A., Koutra, D.: Graph summarization methods and applications: a survey. ACM Comput. Surv. **51**(3), 62:1–62:34 (2018)
60. Louati, A., Aufaure, M., Lechevallier, Y.: Graph aggregation: application to social networks. In: Advances in Theory and Applications of High Dimensional and Symbolic Data Analysis, HDSDA 2011, October 27–30, 2011, Beihang University, Beijing, China, pp. 157–177 (2011)
61. Lucchese, C., Orlando, S., Perego, R.: A unifying framework for mining approximate top-*k* binary patterns. IEEE Trans. Knowl. Data Eng. **26**(12), 2900–2913 (2014)
62. Luo, Y., Fletcher, G.H.L., Hidders, J., Wu, Y., Bra, P.D.: External memory k-bisimulation reduction of big graphs. In: 22nd ACM International Conference on Information and Knowledge Management, CIKM'13, San Francisco, CA, USA, October 27–November 1, 2013, pp. 919–928 (2013)
63. Marketakis, Y., Minadakis, N., Kondylakis, H., Konsolaki, K., Samaritakis, G., Theodoridou, M., Flouris, G., Doerr, M.: X3ML mapping framework for information integration in cultural heritage and beyond. Int. J. Digit. Libr. **18**(4), 301–319 (2017)
64. Milo, T., Suciu, D.: Index structures for path expressions. In: Database Theory—ICDT '99, 7th International Conference, Jerusalem, Israel, January 10–12, 1999, Proceedings, pp. 277–295 (1999)
65. Minadakis, N., Marketakis, Y., Kondylakis, H., Flouris, G., Theodoridou, M., de Jong, G., Doerr, M.: X3ML framework: An effective suite for supporting data mappings. In: Proceedings of the Workshop on Extending, Mapping and Focusing the CRM Co-located with 19th International Conference on Theory and Practice of Digital Libraries (2015), Poznań, Poland, September 17, 2015, pp. 1–12 (2015)
66. Motta, E., Mulholland, P., Peroni, S., d'Aquin, M., Gómez-Pérez, J.M., Mendez, V., Zablith, F.: A novel approach to visualizing and navigating ontologies. In: The Semantic Web—ISWC 2011—10th International Semantic Web Conference, Bonn, Germany, October 23–27, 2011, Proceedings, Part I, pp. 470–486 (2011)
67. Murtagh, F.: A survey of recent advances in hierarchical clustering algorithms. Comput. J. **26**(4), 354–359 (1983)
68. Mynarz, J., Dudás, M., Tomeo, P., Svátek, V.: Generating examples of paths summarizing RDF datasets. In: Joint Proceedings of the Posters and Demos Track of the 12th International Conference on Semantic Systems—SEMANTiCS2016 and the 1st International Workshop on Semantic Change & Evolving Semantics (SuCCESS'16) Co-located with the 12th International Conference on Semantic Systems (SEMANTiCS 2016), Leipzig, Germany, September 12–15, 2016 (2016)
69. Navlakha, S., Rastogi, R., Shrivastava, N.: Graph summarization with bounded error. In: Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2008, Vancouver, BC, Canada, June 10–12, 2008 (2008)
70. Neumann, T., Weikum, G.: The RDF-3X engine for scalable management of RDF data. VLDB J. **19**(1), 91–113 (2010)
71. Paige, R., Tarjan, R.E.: Three partition refinement algorithms. SIAM J. Comput. **16**(6), 973–989 (1987)
72. Palmonari, M., Rula, A., Porrini, R., Maurino, A., Spahiu, B., Ferme, V.: ABSTAT: linked data summaries with abstraction and statistics. In: The Semantic Web: ESWC 2015 Satellite Events—ESWC 2015 Satellite Events Portorož, Slovenia, May 31–June 4, 2015, Revised Selected Papers, pp. 128–132 (2015)
73. Pan, J.Z., Gómez-Pérez, J.M., Ren, Y., Wu, H., Wang, H., Zhu, M.: Graph pattern based RDF data compression. In: Semantic Technology - 4th Joint International Conference, JIST 2014, Chiang Mai, Thailand, November 9–11, 2014. Revised Selected Papers, pp. 239–256 (2014)
74. Pappas, A., Troullinou, G., Roussakis, G., Kondylakis, H., Plexousakis, D.: Exploring importance measures for summarizing RDF/S kbs. In: The Semantic Web—14th International Conference, ESWC 2017, Portorož, Slovenia, May 28–June 1, 2017, Proceedings, Part I, pp. 387–403 (2017)
75. Peroni, S., Motta, E., d'Aquin, M.: Identifying key concepts in an ontology, through the integration of cognitive principles with statistical and topological measures. In: The Semantic Web, 3rd Asian Semantic Web Conference, ASWC 2008, Bangkok, Thailand, December 8-11, 2008. Proceedings, pp. 242–256 (2008)
76. Pham, M., Passing, L., Erling, O., Boncz, P.A.: Deriving an emergent relational schema from RDF data. In: Proceedings of the 24th International Conference on World Wide Web, WWW 2015, Florence, Italy, May 18–22, 2015, pp. 864–874 (2015)
77. Picalausa, F., Luo, Y., Fletcher, G.H.L., Hidders, J., Vansummeren, S.: A structural approach to indexing triples. In: The Semantic Web: Research and Applications—9th Extended Semantic Web Conference, ESWC 2012, Heraklion, Crete, Greece, May 27–31, 2012. Proceedings (2012)
78. Pires, C.E.S., Queiroz-Sousa, P.O., Kedad, Z., Salgado, A.C.: Summarizing ontology-based schemas in PDMS. In: Workshops Proceedings of the 26th International Conference on Data Engineering, ICDE 2010, March 1–6, 2010, Long Beach, California, USA, pp. 239–244 (2010)
79. Pouriyeh, S.A., Allahyari, M., Kochut, K., Arabnia, H.R.: A comprehensive survey of ontology summarization: measures and methods. CoRR arXiv:1801.01937 (2018)
80. Presutti, V., Aroyo, L., Adamou, A., Schopman, B.A.C., Gangemi, A., Schreiber, G.: Extracting core knowledge from linked data. In: Proceedings of the Second International Workshop on Consuming Linked Data (COLD2011), Bonn, Germany, October 23, 2011 (2011)
81. Queiroz-Sousa, P.O., Salgado, A.C., Pires, C.E.S.: A method for building personalized ontology summaries. JIDM **4**(3), 236–250 (2013)
82. Qun, C., Lim, A., Ong, K.W.: D(k)-index: An adaptive structural summary for graph-structured data. In: Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data, San Diego, California, USA, June 9–12, 2003, pp. 134–144 (2003)
83. Riondato, M., García-Soriano, D., Bonchi, F.: Graph summarization with quality guarantees. Data Min. Knowl. Discov. **31**(2), 314–349 (2017)
84. Rissanen, J.: Modeling by shortest data description. Automatica **14**(5), 465–471 (1978)
85. Rudolf, M., Paradies, M., Bornhövd, C., Lehner, W.: Synopsys: large graph analytics in the SAP HANA database through summarization. In: First International Workshop on Graph Data Management Experiences and Systems, GRADES 2013, Co-loated with SIGMOD/PODS 2013, New York, NY, USA, June 24, 2013, p. 16 (2013)
86. Schätzle, A., Neu, A., Lausen, G., Przyjaciel-Zablocki, M.: Large-scale bisimulation of RDF graphs. In: Proceedings of the Fifth Workshop on Semantic Web Information Management, SWIM@SIGMOD Conference 2013, New York, NY, USA, June 23, 2013, pp. 1:1–1:8 (2013)
87. Schmachtenberg, M., Bizer, C., Paulheim, H.: State of the LOD Cloud 2014. http://linkeddatacatalog.dws.informatik.uni-mannheim.de/state/. Accessed 30 Mar 2017
88. Schwarte, A., Haase, P., Hose, K., Schenkel, R., Schmidt, M.: FedX: optimization techniques for federated query processing on linked data. In: The Semantic Web—ISWC 2011—10th Inter-

national Semantic Web Conference, Bonn, Germany, October 23–27, 2011, Proceedings, Part I, pp. 601–616 (2011)
89. Seah, B., Bhowmick, S.S., Dewey, C.F., Yu, H.: FUSE: a profit maximization approach for functional summarization of biological networks. BMC Bioinform. **13**(S–3), S10 (2012)
90. Song, Q., Wu, Y., Dong, X.L.: Mining summaries for knowledge graph search. In: IEEE 16th International Conference on Data Mining, ICDM 2016, December 12–15, 2016, Barcelona, Spain, pp. 1215–1220 (2016)
91. Spahiu, B., Porrini, R., Palmonari, M., Rula, A., Maurino, A.: ABSTAT: ontology-driven linked data summaries with pattern minimalization. In: SumPre (2016)
92. Spahiu, B., Porrini, R., Palmonari, M., Rula, A., Maurino, A.: ABSTAT: ontology-driven linked data summaries with pattern minimalization. In: The Semantic Web—ESWC 2016 Satellite Events, Heraklion, Crete, Greece, May 29–June 2, 2016, Revised Selected Papers, pp. 381–395 (2016)
93. Stefanoni, G., Motik, B., Kostylev, E.V.: Estimating the Cardinality of Conjunctive Queries over RDF Data Using Graph Summarisation. Research report, University of Oxford (2017). https://www.cs.ox.ac.uk/isg/tools/SumRDF/paper-tr.pdf
94. Stocker, M., Seaborne, A., Bernstein, A., Kiefer, C., Reynolds, D.: SPARQL basic graph pattern optimization using selectivity estimation. In: Proceedings of the 17th International Conference on World Wide Web, WWW 2008, Beijing, China, April 21–25, 2008, pp. 595–604 (2008)
95. Sydow, M., Pikula, M., Schenkel, R.: The notion of diversity in graphical entity summarisation on semantic knowledge graphs. J. Intell. Inf. Syst. **41**(2), 109–149 (2013)
96. Tian, Y., Hankins, R.A., Patel, J.M.: Efficient aggregation for graph summarization. In: Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2008, Vancouver, BC, Canada, June 10–12, 2008, pp. 567–580 (2008)
97. Tian, Y., Patel, J.M.: Interactive graph summarization. In: Link Mining: Models, Algorithms, and Applications, pp. 389–409. Springer (2010)
98. Tran, T., Ladwig, G., Rudolph, S.: Managing structured and semistructured RDF data using structure indexes. IEEE TKDE **25**(9), 2076–2089 (2013)
99. Troullinou, G., Kondylakis, H., Daskalaki, E., Plexousakis, D.: RDF digest: efficient summarization of RDF/S kbs. In: The Semantic Web. Latest Advances and New Domains—12th European Semantic Web Conference, ESWC 2015, Portoroz, Slovenia, May 31–June 4, 2015. Proceedings, pp. 119–134 (2015)
100. Troullinou, G., Kondylakis, H., Daskalaki, E., Plexousakis, D.: RDF digest: ontology exploration using summaries. In: Proceedings of the ISWC 2015 Posters & Demonstrations Track Co-located with the 14th International Semantic Web Conference (ISWC-2015), Bethlehem, PA, USA, October 11, 2015 (2015)
101. Troullinou, G., Kondylakis, H., Daskalaki, E., Plexousakis, D.: Ontology understanding without tears: The summarization approach. Semant. Web **8**(6), 797–815 (2017)
102. Troullinou, G., Kondylakis, H., Stefanidis, K., Plexousakis, D.: Exploring RDFS kbs using summaries. In: The Semantic Web—ISWC 2018—17th International Semantic Web Conference, Monterey, CA, USA, October 8–12, 2018, Proceedings, Part I, pp. 268–284 (2018)
103. Troullinou, G., Kondylakis, H., Stefanidis, K., Plexousakis, D.: Rdfdigest+: a summary-driven system for kbs exploration. In: Proceedings of the ISWC 2018 Posters & Demonstrations, Industry and Blue Sky Ideas Tracks Co-located with 17th International Semantic Web Conference (ISWC 2018), Monterey, USA, October 8–12, 2018 (2018)
104. Udrea, O., Pugliese, A., Subrahmanian, V.S.: 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, pp. 1465–1470 (2007)
105. Valiant, L.G.: A bridging model for parallel computation. Commun. ACM **33**(8), 103–111 (1990)
106. W3C: Resource description framework. http://www.w3.org/RDF/
107. W3C: Owl 1 web ontology language. https://www.w3.org/TR/owl-features/ (2012)
108. W3C: Owl 2 web ontology language. https://www.w3.org/TR/owl2-overview/ (2012)
109. W3C: SPARQL 1.1 query language. http://www.w3.org/TR/sparql11-query/ (2013)
110. Wu, G., Li, J., Feng, L., Wang, K.: Identifying potentially important concepts and relations in an ontology. In: The Semantic Web—ISWC 2008, 7th International Semantic Web Conference, ISWC 2008, Karlsruhe, Germany, October 26–30, 2008. Proceedings, pp. 33–49 (2008)
111. Yan, X., Yu, P.S., Han, J.: Graph indexing: a frequent structure-based approach. In: Proceedings of the ACM SIGMOD International Conference on Management of Data, Paris, France, June 13–18, 2004, pp. 335–346 (2004)
112. You, J., Pan, Q., Shi, W., Zhang, Z., Hu, J.: Towards graph summary and aggregation: a survey. In: Zhou, S., Wu, Z. (eds.) Social Media Retrieval and Mining, pp. 3–12. Springer (2013)
113. Zhang, H., Duan, Y., Yuan, X., Zhang, Y.: ASSG: adaptive structural summary for RDF graph data. In: Proceedings of the ISWC 2014 Posters & Demonstrations Track a track within the 13th International Semantic Web Conference, ISWC 2014, Riva del Garda, Italy, October 21, 2014., pp. 233–236 (2014)
114. Zhang, N., Tian, Y., Patel, J.M.: Discovery-driven graph summarization. In: Proceedings of the 26th International Conference on Data Engineering, ICDE 2010, March 1–6, 2010, Long Beach, California, USA, pp. 880–891 (2010)
115. Zhang, X., Cheng, G., Ge, W., Qu, Y.: Summarizing vocabularies in the global semantic web. J. Comput. Sci. Technol. **24**(1), 165–174 (2009)
116. Zhang, X., Cheng, G., Qu, Y.: Ontology summarization based on rdf sentence graph. In: Proceedings of the 16th International Conference on World Wide Web, WWW 2007, Banff, Alberta, Canada, May 8–12, 2007, pp. 707–716 (2007)
117. Zhao, P., Yu, J.X., Yu, P.S.: Graph indexing: tree + delta >= graph. In: Proceedings of the 33rd International Conference on Very Large Data Bases, University of Vienna, Austria, September 23–27, 2007 (2007)
118. Zheng, W., Zou, L., Peng, W., Yan, X., Song, S., Zhao, D.: Semantic SPARQL similarity search over RDF knowledge graphs. PVLDB **9**(11), 840–851 (2016)
119. Zneika, M., Lucchese, C., Vodislav, D., Kotzinos, D.: RDF graph summarization based on approximate patterns. In: Information Search, Integration, and Personalization—10th International Workshop, ISIP 2015, Grand Forks, ND, USA, October 1–2, 2015, Revised Selected Papers, pp. 69–87 (2015)
120. Zneika, M., Lucchese, C., Vodislav, D., Kotzinos, D.: Summarizing linked data RDF graphs using approximate graph pattern mining. In: Proceedings of the 19th International Conference on Extending Database Technology, EDBT 2016, Bordeaux, France, March 15–16, 2016, pp. 684–685 (2016)
121. Zneika, M., Vodislav, D., Kotzinos, D.: Quality Metrics For RDF Graph Summarization. Semant. Web J. (SWJ) (2018) **(accepted, to appear)**