RDF Digest: Efficient Summarization of RDF/S KBs

Georgia Troullinou, Haridimos Kondylakis, Evangelia Daskalaki, Dimitris Plexousakis

Institute of Computer Science, FORTH, N. Plastira 100, Heraklion, Greece

{troulin, kondylak, eva, dp}@ics.forth.gr

Abstract. The exponential growth of the web and the extended use of semantic web technologies has brought to the fore the need for quick understanding, flexible exploration and selection of complex web documents and schemas. To this direction, ontology summarization aspires to produce an abridged version of the original ontology that highlights its most representative concepts. In this paper, we present *RDF Digest*, a novel platform that automatically produces summaries of RDF/S Knowledge Bases (KBs). A summary is a valid RDFS document/graph that includes the most representative concepts of the schema adapted to the corresponding instances. To construct this graph, our algorithm exploits the semantics and the structure of the schema and the distribution of the corresponding data/instances. The performed preliminary evaluation demonstrates the benefits of our approach and the considerable advantages gained.

Keywords: Semantic Summaries, RDF/S documents/graphs, Schema Summary

1 Introduction

The vision of Semantic Web is the creation of a common framework that allows data to be shared and reused across application, enterprise, and community boundaries. Ontologies are playing an important role in the development and deployment of the Semantic Web since they model the structure of knowledge and try to organize information for enhancing the understanding of the contextual meaning of data. Lately, ontologies have been used in database integration [1], obtaining promising results, for example in the fields of biomedicine and bioinformatics, but also as means for publishing large volumes of interlinked data from which we can retrieve abundant knowledge. The Linked Open Data cloud for example contains more than 62 billion triples (as of January 2014).

Given the explosive growth in both data size and schema complexity, data sources are becoming increasingly difficult to understand and use. They often have extremely complex schemas which are difficult to comprehend, limiting the exploration and the exploitation potential of the information they contain. Moreover, regarding ontology engineering, ontology understanding is a key element for further development and reuse. For example, a user/ontology engineer, in order to formulate queries, has to examine carefully the entire schema in order to identify the interesting elements. Besides

adfa, p. 1, 2011.

schema, the data contained in the different sources should also drive the identification of the relevant items. Currently, an efficient and effective way to understand the content of each source without examining all data is still a blind spot.

As a result, there is now, more than ever, an increasing need to develop methods and tools in order to facilitate the understanding and exploration of various data sources. Approaches for ontology modularization [2] and partitioning [3] try to minimize and partition ontologies for better understanding but without preserving the important information. Other works focus on providing overviews on the aforementioned ontologies [5,6,7,8], [13] maintaining however the more important ontology elements. Such an overview can also be provided by means of an ontology summary. Ontology summarization [7] is defined as the process of *distilling knowledge from an ontology in order to produce an abridged version*. While summaries are useful, creating a "good" summary is a non-trivial task. A summary should be concise, yet it needs to convey enough information to enable a decent understanding of the original schema. Moreover, the summarization should be coherent and provide an extensive coverage of the entire ontology (multiple subjects of the ontology). So far, although a reasonable number of research works tried to address the problem of summarization from different angles, a solution that simultaneously exploits the semantics provided by the schemas and the data instances is still missing.

In this paper, we focus on RDF/S ontologies and demonstrate an efficient and effective method to automatically create high-quality summaries. A summary constitutes a "valid" sub-schema providing an overview of the original schema considering also the available data. Specifically the contributions of this paper are the following:

- We present RDF Digest, a novel platform that automatically produces RDF schema summaries that highlight the most representative concepts of the schema adapted to the corresponding data instances.
- In order to construct these summarized graphs our system exploits a) the *semantics* of the schema, b) the *structure* of the RDFS graph and c) the *distribution* of the corresponding *data/instances* in order to identify and select the most important and representative elements of the ontology.
- To identify the most important nodes we define the notion of *relevance* based on the *relative cardinality* and the *in/out degree centrality* of a node. Moreover, to ensure that our summary selects the most representative nodes of the entire schema we use the notion of *coverage*. Those two notions are combined in an algorithm that finally produces a "valid" summary schema out of the original schema.
- Finally, our experimental evaluation show the feasibility of our approach and the considerable advantages gained.

To our knowledge, this is the first approach that, in the context of ontology, combines both schema and data instance information to produce a high-quality summary graph.

The rest of the paper is organized as follows. Section 2 introduces the formal framework of our solution and Section 3 describes the metrics used in our algorithms to determine the nodes and paths to be included in the summary. Section 4 presents our algorithm and Section 5 describes the evaluation conducted. Section 6 presents related work and finally, Section 7 concludes the paper.

2 Preliminaries

Schema summarization aims to highlight the most representative concepts of a schema, preserving "important" information and reducing the size and the complexity of the schema [8]. Despite the significance of the problem there is still no universally accepted measurement on the importance of nodes in an RDF/S graph. In our approach, we try to elicit this information from a) the *structure* of the graph, b) the *semantics* of the schema and c) the *distribution* of the corresponding *data*. Our goal is to produce a simple and expressive graph that presents an overview of the schema and also provides an intuition about the corresponding stored data.

Specifically, in this paper we focus on RDF/S KBs, as RDF/S is the de-facto standard for publishing and representing data on the web [9]. The representation of knowledge in RDF is based on triples of the form of *(subject predicate object)*. RDF datasets have attached semantics through RDF Schemas [10]. RDF Schema is a vocabulary description language that includes a set of inference rules used to generate new, implicit triples from explicit ones. Note that in our case the inference is implemented only at the RDF schema level to avoid overloading the super-classes with instances. Each RDF schema S defines a finite set of class names C and property names P. Properties are defined using class names or literal types, so that, for each property p, the domain of property p, i.e. *domain(p)*, is a class and the range of p, i.e. *range(p)*, is either a class or a literal. The classes and the properties of a schema are uniquely identified by the names in $N = C \cup P$ (possibly using namespace URIs for disambiguation). Moreover, we denote by $H = (N, \prec)$, a hierarchy of class and property names. H is well-formed if $\prec$ is a smallest partial ordering such that: if $p1, p2 \in P$ and $p1 \prec p2$, then $domain(p1) \prec domain(p2)$ and $range(p1) \prec range(p2)$. In this paper, we ignore unnamed resources, also called blank nodes. Moreover, for the representation of the RDF/S documents we will use a graph data model first introduced by Karvounarakis et. al [11]. Formally, we define an RDF schema graph as:

Definition 1 (RDF schema graph)*:* An *RDF schema graph* S is a labeled directed graph $S = (V, E, \lambda_c, \lambda_p, H)$ depicting a collection of triples $T_S = (s, p, o) = URIs \times URIs \times URIs$ where:

- V represents a set of nodes.
- E represents a set of edges of the form $e(v_i, v_j)$ with $v_i, v_j \in V$ and direction from v_i to v_j. Given that, e, v_i, v_j correspond to a property p, the *domain(p)* and the *range(p)*, respectively. The label of e is $\lambda_P(e) = p$, where $p \in P$.
- H is a well-formed hierarchy of a class and property names $H = (N, \prec)$
- λ_c: is a value function that assigns to each node $v \in V$ in S a class name (URI) from C. Such as $\lambda_c(v) = c$, $c \in C$.
- λ_p: is a value function that assigns to each edge $e \in E$ in S one property name from P. Such as $\lambda_p(e) = p$, $p \in P$.

Moreover, we assume a function κ_P that characterizes the type of a property p among the *standard RDF properties* (e.g. *"rdfs:subClassOf", "rdfs:label"*) and the *user defined properties*. RDF schema provides also *inference semantics*, which is of two types,

namely *structural inference* (provided mainly by the transitivity of subsumption relations) and *type inference* (provided by the typing system, e.g., if *p* is a property, the triple {*p*, *type*, *property*} can be inferred). The RDF schema, which contains all triples that are either explicit or can be inferred from explicit triples in an RDF graph *S* (using *both* types of inference), is called the *closure* of *S* and is denoted by *Cl(S)*. An *RDFS KB S* is an RDF schema graph, which is closed with respect to *type inference*, i.e., it contains all the triples that can be inferred from *S* using type inference. We also assume that the RDF/S KBs are *valid*. The validity constraints that we consider concern *type uniqueness*, i.e., each resource has a unique type, the *acyclicity* of the *subClassOf* and *subPropertyOf* relations and that the subject and object of the instance of some property should be correctly classified under the domain and range of the property, respectively. The full list of the validity constraints we adopt is contained in [12]. Those constraints are enforced to enable unique and non-ambiguous detection of the summary. Next, we define an RDF instance graph.

Definition 2 (RDF instance graph)*: An RDF instance graph I*, is a labeled directed graph $I = (N, R, \tau_v, \tau_c, \tau_p)$, depicting a collection of triples $T_I = (s, p, o) = URIs$ x $URIs$ x ($URIs \cup Literals$) where:

- *N* is a finite set of nodes.
- *R* is a finite set of directed edges between nodes, $r(n_i, n_j)$ with $n_i, n_j \in N$ and direction from n_i to n_j.
- τ_v: is a value function that assigns to each node $n \in N$ in *I* a *URI* or a *literal*.
- τ_p: is a value function that correlates edges of *S* to edges of *I*. (such that $\tau_p(r) = \lambda_p(e)$). For each edge *r* in *R* going from a node n_i to a node n_j, τ_p returns a property name $p \in P$, where values n_i and n_j belongs to the interpretation of *p*: $domain(p) = \lambda_c(v_i) = \tau_c(n_i)$, $range(p) = \lambda_c(v_j) = \tau_c(n_j)$
- τ_c: is a labeling function that captures *rdfs:type* declarations, linking the RDF instance graph *I* with the RDF schema graph *S*. The τ_c returns either the name of a class $c \in C$ or the value of the container type (literal). Based on terms of RDFS, the $n \in N$ is an instance of class $\lambda_c(v)$ (or the *n* is type class $\lambda_c(v)$), where $v \in V$.

Now, as an example, consider the CIDOC-CRM[1] ontology part shown in Fig. 1 used to describe the process of information acquisition and the involved actors in cultural heritage. Although this is only a short example, we have 27 classes and many properties that need to be examined in order to understand the schema. In blue color, we can see the summarized graph as it is produced by our method. Obviously, it is easier to understand schema content using only the summary graph since it contains the most important nodes out of the initial graph.

3 Assessment Measures

In this section, we present the properties that a sub-graph of our schema is required to have in order to be considered a high-quality summary. Specifically, we are interested in important schema nodes that can describe efficiently the whole schema and reflect

[1] http://www.cidoc-crm.org/official_release_cidoc.html

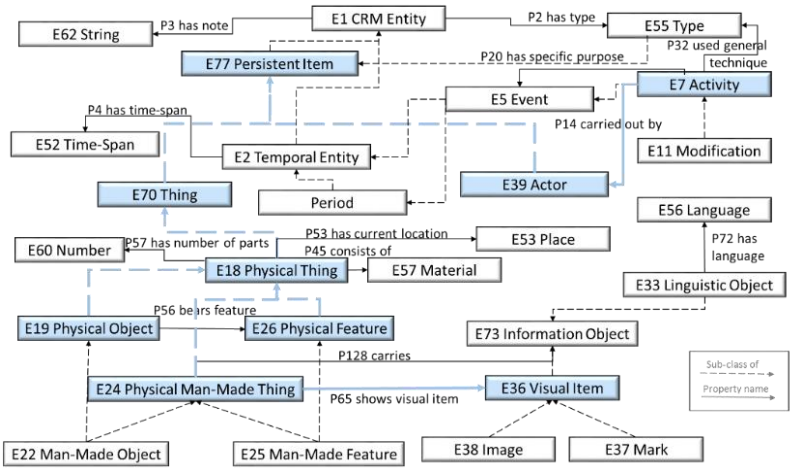


Fig. 1. Example of RDFS Knowledge Base and the corresponding summary graph (in blue)

the distribution of the data instances at the same time. To capture these properties, we use the notions of *relevance* and *coverage* which are further analyzed below. Relevance is used for identifying the most important nodes and coverage is used for extracting nodes/paths, which cover the whole spectrum of the RDF/S document.

3.1 Relevance

Importance has a broad range of meanings and this has led to many different algorithms that try to identify it. Originating from the analysis of social graphs, in the domain of Semantic Web, algorithms adapting the well-known PageRank [4,5] have been proposed to determine the importance of elements in an XML document. For RDF/S, other approaches use measures such as the *degree centrality*, the *between-ness* and the eigenvector centrality *(weighted Page Rank and HITS)* [7], adjusting them to the specific features of RDF/S or they try to adapt *the degree* centrality and *the closeness* [8] to calculate the relevance of a node.

In our case, we believe that the importance of a node should describe how well a node could represent a part of a KB (its area) giving an intuition about its neighborhood. Intuitively, nodes with many connections in a schema graph will have a high importance. However, since RDF/S KBs might contain huge amounts of data, the latter data should also be involved when trying to estimate a node's importance.

Consider for example the node *"E37 Mark"* and the node *"E38 Image"* in the schema graph of Fig. 1. The two nodes have the same number of connections and they are connected to the same node *"E18 Physical Thing"*. Now assume that the node *"E38 Image"* has the double number of instances. Due to the same number of connections, the two nodes may be considered equal but essentially the *"E38 Image"* is more important for the specific RDF/S KB, due to the higher number of instances it contains. Obviously, the number of instances of the class - that a node corresponds to - is a valuable piece of information for identifying its importance.

In our approach, initially, we determine how central/important a node is, judging from the instances it contains (*relative cardinality*). After that, we estimate the centrality of a node in the entire KB (*in/out centrality*), combining the relative cardinality with the number and type of the incoming and outgoing edges in the schema. Finally, the *relevance* of a schema node is defined by comparing its centrality with the centrality of its neighbors.

Relative Cardinality. The cardinality of a schema node is the number of instances it contains in the current RDF/S KB. If there are many instances of a specific class, then that class is more likely to be more important than another with very few instances. Similarly, the cardinality of an edge between two nodes in a graph is the number of the corresponding instances of the nodes connected with that specific edge. Now we can formally define the relative cardinality of an edge.

Definition 3 (Relative Cardinality of an edge). Let $S = (V, E, \lambda_c, \lambda_p, H)$ be an RDF schema graph and $I = (N, R, \tau_v, \tau_c, \tau_p)$ the RDF instance graph of S. The *relative cardinality* of an edge $e(v_i, v_j)$ in S, where $e \in E$ and $v_i, v_j \in V$, i.e. the $RC(e(v_i, v_j))$, (remember that $\lambda_p(e) = p$) is the following:

- *In case of available instances*: The number of specific instance connections $r(n_i, n_j) \in R$, n_i , $n_j \in N$, where $\tau_p(r) = \lambda_p(e)$, $\tau_c(n_i) = \lambda_c(v_i)$ *and* $\tau_c(n_j) = \lambda_c(v_j)$, divided by the total number of the connections ($r_k(n_i, n_a)$, $r_t(n_b, n_j) \in R$, *where* $n_a, n_b \in N$) of the instances of these two nodes v_i, v_j . A constant value a is added to this number.
- *In case of no available instances*: A constant value a.

$$RC\left(e\left(v_i, v_j\right)\right) = \begin{cases} \alpha + \dfrac{\left|\left\{r_m\left(n_i, n_j\right)\right\}\right|}{\left|\left\{r_k\left(n_i, n_a\right)\right\}\right| + \left|\left\{r_t\left(n_b, n_j\right)\right\}\right|}, & r_m\left(n_i, n_j\right) \in R \\ \alpha, & r_m\left(n_i, n_j\right) \notin R \end{cases} \quad (1)$$

The constant value a has the value 1/#*connections* where #*connections* is the number of connections $e(v_i, v_j)$ that exist in the schema. Our algorithm is flexible enough to focus on the available instances when they exist, and if they are not available, it only exploits the semantics and the structure of the schema.

In/Out Centrality. In order to combine the notion of centrality in the schema and the distribution of the corresponding dataset, we define the *in/out centrality*, exploiting also the relative cardinality of nodes and edges. The *in/out* centrality is an adaptation of *the degree centrality* [7]. In an undirected graph, the *degree centrality* is defined as the number of links incident upon a node. In a directed graph however, as in our case, the degree centrality is distinguished to the *in-degree centrality* and the *out-degree centrality*.

The *in-centrality* of a schema node v, i.e. $C_{in}(v)$, is the sum of the weighted relative cardinalities of the incoming edges. The weights, that are used, are experimentally defined and depend on the types of the properties as they are identified by the function κ_P.

As already mentioned, there are two types of properties, the standard RDF types (for example "*rdfs:subClassOf*", "*rdfs:label*", "*rdfs:comment*") and the user defined properties (for example the *"P45 consists of"*, *"P128 carries"* shown in Fig. 1). We would like to consider as more important the latter, whereas the former are not considered to be equally important. This is partly because the user-defined properties correlate classes, each exposing the connectivity of the entire schema, in contrast to the hierarchical RDF/S properties.

Definition 4 (in(out)-centrality of a node). Let *S* be an RDF schema graph and *m* be the number of the incoming (outgoing) edges $e(v_i , v)$ $(e(v , v_i))$ of a node *v* in S. The $C_{in}(v)$ $(C_{out}(v))$ of *v* is the sum of the *relative cardinality* of the edges $e(v_i , v)$ $(e(v , v_i))$, multiplied by a weight w_p according to the type of edge.

$$C_{in}(v) = \sum_{1}^{m} RC(v_i, v) * w_p \qquad C_{out}(v) = \sum_{1}^{m} RC(v, v_i) * w_p \tag{2}$$

Relevance. The notion of centrality, as defined previously, is a measure that can give us an intuition about how central a schema node in an RDF/S KB is. However, its importance should be determined considering also the centrality of the other nodes as well. Consider for example, the nodes *"E60 Number"* and *"E56 Language"* shown in Fig. 1. They have the same number of incoming and outgoing edges and assume that they have the same number of instances as well. However the *"E60 Number"* is connected to more important elements compared to the *"E56 Language"*. For example, the node *"E18 Physical Thing"* is directly connected to the *"E60 Number"* and has many other connections and instances. Since the *"E18 Physical Thing"* is obviously a very important node, the *"E60 Number"* is a less appropriate node to represent this area in a summary. On the other hand, the *"E56 Language"* is more relevant than the *"E60 Number"* to represent the specific part of the graph since its neighbors do not have such a high relevance.

To achieve the aforementioned goal, the *relevance* of a node is affected by its surrounding neighbors and more specifically by the number and the connections of its adjacent nodes. To be more precise, the formula estimates the (number of) connections of a node and this number is compared to the connections of its neighbors.

Definition 5 (Relevance of a node). Let *S* be an RDF schema graph**,** np_{in} be the number of incoming nodes v_i connected to *v* with $e_a(v_i, v)$, and the np_{out} be the number of outgoing nodes v_j connected to *v* with $e_b(v, v_j)$. The *relevance* of *v*, i.e. *Relevance(v)*, is the sum of *in* and *out centrality* of *v* multiplied by the corresponding number of nodes, divided by the sum of *out-centrality* of the incoming nodes v_i and the *in-centrality* of the outgoing nodes v_j.

$$\mathrm{Re}levance(v) = \frac{C_{in}(v) * np_{in} + C_{out}(v) * np_{out}}{\sum_{1}^{np_{in}} (C_{out}(v_i)) + \sum_{1}^{np_{out}} (C_{in}(v_j))} \tag{3}$$

Obviously, the relevance of a schema node in an RDF/S KB is determined by both its connectivity in the schema and the cardinality of the instances. Thus, the number of instances of a node is of vital importance in the assessment procedure. When the data distribution significantly changes, the focus of the entire data source is shifted as well, and as a result, the relevance of the nodes changes. In addition, the importance of each

node is compared to the other nodes in the specific area/neighborhood in order to identify the most relevant nodes that can represent all the concepts of a graph.

3.2 Coverage

After having estimated the relevance of each node in the schema graph, it is now time to focus on the paths that exist in a schema graph. The idea behind this is that we are not interested in extracting isolated nodes, but most importantly we want to produce valid sub-schema graphs. So the chosen paths should be selected having in mind to collect the more relevant nodes by minimizing the overlaps.

Definition 6 (Path $v_s \longrightarrow v_i$). A *path* from v_s to v_i, i.e. $v_s \longrightarrow v_i$, is the finite sequence of edges, which connect a sequence of nodes, starting from the node v_s and ending in the node v_i.

As a consequence, the relative cardinality of a path is the sum of relative cardinalities of the individual edges. Moreover, the length of a path, i.e. $d_{vs \rightarrow vi}$, is the number of the edges that exist in that path.

In our running example of Fig. 1, the nodes "*E53 Place*" and "*E57 Material*" are directly connected to the node "*E18 Physical Thing*" and have similar connectivity in the graph. The node "*E18 Physical Thing*" has a high relevance in the graph and as a consequence a great probability to be included in the summary. However, although the "*E18 Physical Thing*" can be located only in one "*E53 Place*", it might consist of many "*E57 Material*". As a consequence, the relative cardinality of the path from the "*E18 Physical Thing*" to the "*E57 Material*" (*RC(e*("*E18 Physical Thing*", "*E57 Material*"))) will be higher than the relative cardinality of the path form "*E18 Physical Thing*" to "*E53 Place*". This means that the path from "*E18 Physical Thing*" to "*E57 Material*" is more representative to be included in the summary than the path from "*E18 Physical Thing*" to "*E53 Place*". This is because the "*E18 Physical Thing*" already covers the "*E53 Place*" - a physical thing is located only in one place.

In the above example, we dealt with paths of length one. However, the paths included in the summary should contain the most relevant schema nodes which represent the remaining nodes, achieving the digest of the entire content of the RDF/S KB. As a consequence, the main criteria to estimate the level of coverage of a specific path are: a) the relevance of each node contained in the path, b) its relevant instances in the dataset and c) the length of the path. As a result, similar to the approach of Yu et al. [5], we define the notion of c*overage* as follows:

Definition 7 (Coverage of a path). Let *S* be an RDF schema graph and *I* be an instance of *S*. The *coverage* of a path $v_s \longrightarrow v_i$, i.e. the *Coverage*($v_s \longrightarrow v_i$), is derived by the sum of the *Relevance* of the sequential nodes v_j contained between the nodes v_s and v_i, multiplied by the *relative cardinality* of each edge $e(v_{j-1}, v_j)$ contained in the path. The result is divided by the length of the path in order to penalize the longer paths.

$$Coverage(v_s \rightarrow v_i) = \frac{1}{d_{n_s \rightarrow n_i}} * \sum_{j=2}^{d_{n_s \rightarrow n_i}} \left(\mathrm{Re}\, levance(v_j) * RC\left(e\left(v_{j-1}, v_j\right)\right)\right) \quad (4)$$

The above formula assesses a path and provides a metric to identify the degree of the contained relevant nodes and how this path can represent (a part of) the original

graph without overlapping issues. Our goal is to select the schema nodes that are more relevant while avoiding having nodes (or paths) in the summary which cover one another. The highest the coverage of a path, the more relevant this path is considered in representing the original graph or part of it.

4 Construction of RDF Summary

Now that we have explained all formulas required in order to calculate the relevance and the coverage of the elements of an RDF/S KB, we can describe the algorithm for constructing the RDF schema summary, shown in Fig. 2. Below we explain in more detail each of the steps of the algorithm.

Algorithm 1: *ComputeRDFSchemaSummary*(*B*, *n*)
Input: An RDF/S Knowledge Base *B*, *n* the number of the requested nodes
Output: An RDF Schema Summary *S*
1. Let *V* be the set of nodes in *B*
2. *for each* node $v_i \in V$
3. r_i := *calculate_relevance*(*B*, v_i)
4. *TOP* := *select_top_nodes*(*B*, *r*, *n*)
5. *ADJ* := *identify_adjacent_nodes*(*TOP*)
6. *S* := *construct_subgraph*(*ADJ*)
7. if *ADJ* < > TOP *then*
8. *for each* node $v_i \in$ *TOP/ADJ*
9. *ADJ* := *ADJ* - v_i
10. *TOP:= TOP* $\cup$ v_i
11. S := S $\cup$ *identify_path_with_max_coverage*(*B*, *S*, v_i)
12. Return *S*

Fig. 2. The algorithm for computing the RDF Schema Summary

In the beginning (lines 2-3) the relevance of each schema node is assessed. Specifically, a value is assigned to each node in the RDF graph according to the *Relevance* measure (calculated using the Def. 5). Having calculated the relevance of each node we would like to get the *n* most important ones to be further elaborated (line 4). Usually *n* is defined by the user. However, if it is left blank this function automatically retrieves a specific percentage of the nodes in the schema (usually 30%-40%). The schema nodes in *TOP* are the structural components to build the schema summary. However, these nodes might not be directly connected in the RDF schema. Since our goal is to create a valid summary schema, we should find the appropriate paths that connect the non-adjacent nodes of the selected collection (lines 5-6). If all nodes are adjacent then the schema summary *S* is the connected subgraph containing these nodes produced using the *construct_subgraph* function. Usually however, the nodes included in the *TOP* set are not adjacent. Nevertheless, they should also be included in the produced summary. The goal is to find paths, which connect these nodes with the already connected ones (lines 7-11). However, we are not looking for random paths but the ones maximizing the coverage. In other words, we select the paths which contain the most relevant nodes according to the coverage measure as described in the previous section. Note that the selection of the nodes to complete the subgraph is done out of the initial RDF schema

graph, since the summary should be coherent with the original schema. Moreover, in this selection, other nodes might be also included in the summary in order to connect the most important ones.

When the algorithm finishes its execution, the selected sub-graph *S*, according to the previous steps, will be the RDF schema summary. In addition, the result of our algorithm for a specific input is unique. If the data distribution changes, the summary is also changed in order to provide an updated view on the corresponding schema and the updated data instances.

5 Evaluation

The algorithm described in this paper was implemented in the RDF Digest prototype. We developed the RDF Digest using JAVA and a beta version of the platform is currently available as a service online[2]. A user can upload the RDF/S document, he would like to be summarized and he is optionally able to define the expected length of the summary as well. When the input is submitted, the RDF/S document is preprocessed by computing the corresponding RDF/S KB. The result is stored in a Virtuoso Instance (http://virtuoso.openlinksw.com/) which enables efficient data access. Then, the algorithm described in Section 4 runs and the results are presented to the user.

To evaluate our system, we selected four ontologies: the BIOSPHERE ontology[3], the Financial ontology[4], the Aktors Portal ontology[5] and the CIDOC-CRM[6] ontology. BIOSPHERE (87 classes, 3 properties) models information in the domain of bio-informatics, the Financial ontology (188 classes, 4 properties) incudes classes and properties in the financial domain and the Aktors Portal ontology (247 classes, 327 properties) describes an academic computer science community. Finally, the CIDOC-CRM (82 classes, 539 properties) provides definitions and a formal structure for cultural heritage documentation. The first three ontologies have been previously used to evaluate relevant works on RDF/S summarization, so we can compare our results with these works. More specifically our algorithms are compared to the algorithms proposed by Peroni et al. [13] and by Queiroz-Sousa et al. [8]. Peroni et al. automatically define the key concepts in an ontology, combining cognitive principles, lexical and topological measurements. Queiroz-Sousa et al. on the other hand propose an algorithm that produces an ontology summary in two manners: automatically using relevance measures and semi-automatically, using the users' opinion in addition. Moreover, we tried but could not get access to [7] to perform the same experiments.

Note that in order to compare our results with the aforementioned works we used only the RDF schema graph of each ontology since the other approaches do not consider instances. To demonstrate a scenario where instances are available we evaluated our

[2] http://www.ics.forth.gr/isl/rdf-digest
[3] http://www.aiai.ed.ac.uk/project/biosphere/downloads.html
[4] http://www.larflast.bas.bg/ontology
[5] http://www.daml.org/ontologies/322
[6] http://www.cidoc-crm.org/official_release_cidoc.html

algorithms using CIDOC-CRM with instances as well. Those instances are real instances retrieving from a real database. Thus, the evaluation is more objective rather than the creation of synthetic data which may not correspond to a real situation. To proceed with the evaluation of the first three ontologies, summaries were generated by eight human experts. These human experts had a good experience in ontology engineering [13] and were familiar with the aforementioned ontologies. The experts were requested to select up to 20 concepts which were considered as the most representative of each ontology. The generated reference summaries were also used by Queiroz-Sousa et al. [8] in their evaluation. The level of agreement among experts for the three ontologies had a mean value of 74% [13] meaning that the experts did not entirely agree on their selections. For CIDOC-CRM, the CIDOC Core[7] ontology was proposed by experts as the core subset of the ontology aimed to represent the basic concepts of CIDOC-CRM into a simple ontology of 29 classes. We used this subset as the reference summary of CIDOC-CRM.

Metrics like precision, recall and F-measure, used by the previous works [8], [13], [14], [15], are limited in exhibiting the added value of a summarization system because of the "disagreement due to synonymy" [16] meaning that they fail to identify closeness with the ideal result when the results are not exactly the same with the reference ones. On the other hand, content based metrics compute the similarity between two summaries in a more reliable way [7]. In the same spirit, Maedche et al. [17] argue that ontologies can be compared at two different levels: lexical and conceptual. At the lexical level, the classes and the properties of the ontology are compared lexicographically, whereas at the conceptual level the taxonomic structures and the relations in the ontology are compared. To this direction, we use the following similarity metric *Sim(S, A)* in order to define the level of agreement between an automatically produced summary *S* and a reference summary *A*.

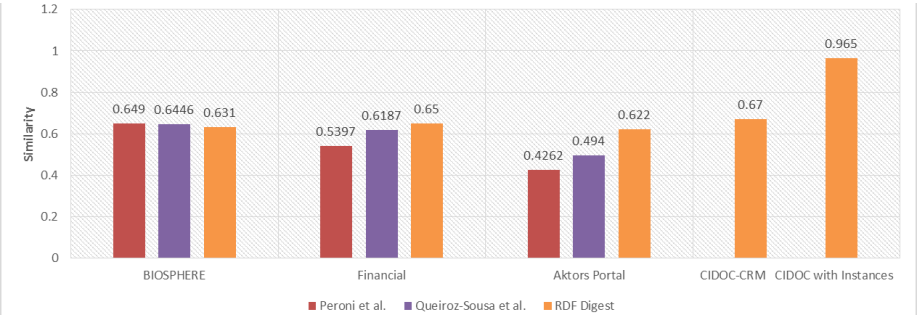
$$Sim(S, A) = \frac{|K_{match}| + 0.6 * \sum_{1}^{|Ksub|} \frac{1}{depth} + 0.3 * \sum_{1}^{|K\,super|} \frac{1}{depth}}{|K|} \quad (5)$$

More precisely, K is set of classes contained in A, $K_{match} \subseteq K$, is the set of classes appearing also in S, $K_{sub} \subseteq K - K_{match}$ / $K_{super} \subseteq K - K_{match}$ (where $K_{sub} \cap K_{super} = \emptyset$) is the set of classes having sub-classes / super-classes in K and *depth* is the distance between the ideal class and the class identified by the summary. Note that the above formalism assesses the existence of sub-classes and the super-classes of S in A with a different percentage. The idea behind that is that the super-classes, since they generalize their sub-classes, are assessed to have a higher weight than the sub-classes. Consequently, the effectiveness of a summarization system is calculated by the average number of the similarity values between the summaries produced by the system and the set of the corresponding experts' summaries. In our case, each summary contains approximately the same number of classes according to the experts' selections, 20 classes for the BIOSPHERE, the Financial, and the Aktros Portal ontologies, and 29 classes for the

[7] http://www.cidoc-crm.org/technical_papers.html

CIDOC-CRM ontology. Our evaluation compares the similarity – as defined previously- between the summaries produced by our algorithm and the reference summaries used by the other works and the results are shown in Fig. 3.

As we can observe, the summaries generated by our system appear to be quite similar to what experts have produced, in most of the cases showing better results than other similar systems. Specifically, the summary of the CIDOC-CRM ontology presents the highest similarity. On the other hand, the results of our system have a good similarity with the experts in the cases of the BIOSPHERE, the Financial, and the Aktros Portal ontologies. We have to note that whereas the reference summaries on these three ontologies contain only isolated classes in the case of CIDOC-CRM the CIDOC Core contains an entire sub-ontology similar to the result we get from our system. This is also the reason for the better results that appear for CIDOC-CRM. Obviously, when



instances are used the similarity of the result summary highly increases and in our case we reach a similarity level of 0.965 which demonstrates the added value of our approach.

Moreover, our system seems to react better when it deals with dense schemas, which are confirmed also by the results shown in Fig. 3. As we can see in the image the Aktors Portal and the CIDOC-CRM ontologies have better results compared to the BIOSPHERE and the Financial ontologies which contain only hierarchical relationships. However, this observation is to be further verified with more experiments.

Furthermore, during our experiments, we observed that as the ontology size and as a consequence the complexity increases, the similarity of the summaries produced by the RDF Digest is improved. This is also depicted in Fig. 4 showing that the similarity increases as the number of properties and classes in the ontology increases as well.

Finally, to test the efficiency of our system, we measured the average time to produce

Fig. 3. A comparative result of ontology summarization methods

the summaries using the aforementioned ontologies. We have to note that the experiments run on a 64 bit Windows 8.1 system with 4GB of main memory and a Core i5 Intel CPU running at 1.6 GHz. The results are shown in Fig. 5. As we can observe, our algorithms produce the requested summary quite fast and require at most 33 sec. Moreover, it is obvious that the larger and the more complex the ontology, the more time it requires to calculate the corresponding RDF schema summary which is reasonable as

it has to calculate the relevance for more nodes and has to perform more path constructions for calculating the coverage.

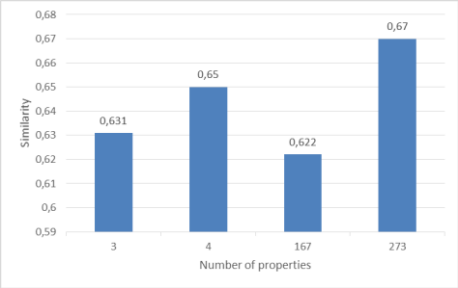


Fig. 4. The similarity as the number of properties increase (CIDOC-CRM)

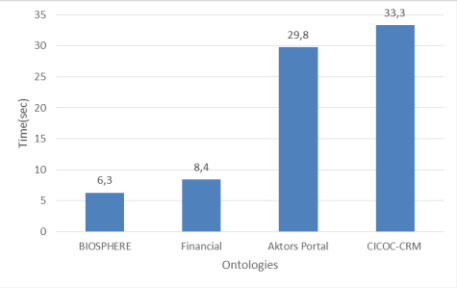


Fig. 5. Execution times for producing the RDF Schema summary

6 Related Work

As already stated, various techniques have been developed for the identification of summaries over different types of schemas and data. The first works on schema summarization focused on conceptual [18] and XML schemas [4,5]. Yu et al. [5] affirm that, while schema structure is of vital importance in summarization, data distribution often provides important knowledge that improves the summary quality. Another work [4] on XML Schemas derives a summary of the schema and then transforms the instances through summary functions. Other works focus on summarizing meta-data and large graphs. For example, Hasan [15] proposes a method to summarize the explanation of the related metadata over a set of Linked Data, based on user specified filtering criteria and producing rankings of explanation statements. One of the latest approaches that deals with graph summaries [19] examines only the structure of an undirected graph, neglecting any additional information (such as semantics). The goal of this work is to generate a summary graph that minimizes the loss of information out of the original graph. However, our system differs from the above in terms of both goals and techniques. Although we reuse interesting ideas from these works, our approach is focused towards RDF/S KBs expressing richer semantics than conceptual schemas and XML.

More closely related works to our data model and approach are [7], [13] and [8]. Zhang et al. [7] propose a method for ontology summarization based on the RDF Sentence Graph. The notion of RDF Sentence is the basic unit for the summarization and corresponds to a combination of a set of RDF statements. The creation of a sentence graph is customized by the domain experts who provide as input the length of the summary and their navigation preferences to create the RDF Sentence graph. The importance of each RDF sentence is assessed by determining its centrality in the graph. In addition, the authors compare five different centrality measures (degree, between-ness, PageRank, HITS), showing that weighted in-degree centrality and some eigenvector-based centralities are better. However, in this approach, the coverage of the entire graph is not considered and many important nodes may be left out.

On the other hand, Peroni et al. [13] try to identify automatically the key concepts in an ontology, combining cognitive principles, lexical and topological measurements such as density and the coverage. The goal is to return a number of concepts that match as much as possible those produced by human experts. However, this work focuses only on hierarchical relationships ignoring the complexity of a graph. In the same direction, Queiroz-Sousa et al. [8] propose an algorithm which produces an ontology summary in two ways: automatically, using relevance measures and, semi-automatically, using additionally the users' opinion (user-defined parameters), producing a personalized ontology summary. However, this work ignores the coverage of the graph thus producing summaries which include nodes that are already represented by other nodes.

Pires et al. [14], propose an automatic method to summarize ontologies that represent schemas of peers participating in a peer-to-peer system. In order to determine the relevance of a concept, a combination two measures, *centrality* and *frequency* is used.

Although in most of works the importance of each node is calculated considering each node in isolation, in our work, we assess its importance in comparison with its neighbors, producing a better result. Moreover, many of these works (such as [8] and [15]) do not consider the coverage of each node and end up collecting nodes already represented by other nodes. In addition, some of these works (e.g. [8], [13]) provide a list of the more important nodes, whereas others [7], [8], [14] and our approach, create a valid summary schema. Finally, other approaches try to navigate on the Linked Data Cloud using summaries of interlinked datasets [20]. However, our work is the only one that automatically produces a summary graph, exploiting the data instances and essentially provides an overview of the entire KB (both schema and instances).

7 Conclusions and Future Work

In this paper, we present a novel method that automatically produces summaries of RDF/S KBs. To achieve that, our algorithm exploits the semantics and structure of the schema and the distribution of the data by combining all these information using the relevance and the coverage properties. The performed evaluation verifies the feasibility of our solution and demonstrates the advantages gained by efficiently producing good summaries. Compared to other similar systems, our approach produces better results, further improved by exploiting knowledge about the instance distribution. Moreover, although most of the systems just select nodes or paths as the result summary, our result is a valid RDFS graph/document out of the initial RDF schema graph and can be used for query answering as well.

We plan to extend our implementation in order to produce the schema summary of large schemas in the Linked Data Cloud. Instead of relying on reference summaries for the evaluation of the automatically produced summaries, an interesting idea is to check if these summaries are able to answer the most common queries formulated by the users. Another interesting topic would be to extend our approach for OWL ontologies. As the size and the complexity of schemas and data increase, ontology summarization is becoming more and more important and several challenges arise.

Acknowledgments. This work was partially supported by the EU projects DIACHRON (FP7-601043), iManageCancer (H2020-643529), MyHealthAvatar (FP7-600929) and EURECA (FP7-288048).

References

1. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Poggi, A., Rodriguez-Muro, M., Rosati, R.:Ontologies and Databases: The DL-Lite Approach, Reas. Web, 255-356 (2009)
2. Stuckenschmidt, H., Parent, C., Spaccapietra, S. (Eds.): Modular ontologies: concepts, theories and techniques for knowledge modularization, 5445, Springer (2009)
3. Stuckenschmidt, H., Klein, M.: Structure-based partitioning of large concept hierarchies. ISWC, pp. 289-303, Springer Berlin Heidelberg (2004)
4. Marciniak, J.: XML Schema and Data Summarization. Artificial Intelligence and Soft Computing. 556-565 (2010)
5. Yu, C., Jagadish, H.V.: Schema Summarization, VLDB, pp. 319-330 (2006)
6. Graves, A., Adali, S., Hendler, J.: A Method to Rank Nodes in an RDF Graph. ISWC (2008)
7. Zhang, X., Cheng, G., Qu, Y.: Ontology Summarization Based on RDF Sentence Graph. WWW, pp. 707-716 (2007)
8. Queiroz-Sousa, P. O., Salgado, A. C., Pires, C. E.: A Method for Building Personalized Ontology Summaries. Journal of Information and Data Management, 4, 3, 236. (2013)
9. Schmachtenberg, M.,Bizer, C.,Paulheim H.: State of the LOD Cloud: http://linkeddatacatalog.dws.informatik.uni-mannheim.de/state/, (November 2014).
10. RDF Schema 1.1 Available online: http://www.w3.org/TR/rdf-schema/, (November 2014).
11. Karvounarakis, G., Alexaki, S., Christophides, V., Plexousakis, D., Scholl, M.: RQL: a declarative query language for RDF, WWW, pp. 592-603 (2002)
12. Serfiotis, G., Koffina, I., Christophides, V., Tannen, V.: Containment and Minimization of RDF/S Query Patterns, ISWC, pp. 607-623 (2005)
13. Peroni, S., Motta, E., and Aquin, M.: Identifying Key Concepts in an Ontology, Through the Integration of Cognitive Principles with Statistical and Topological Measures. In The Semantic Web Journal. 5367, 242–256. Berlin, Germany (2008)
14. Pires, C. E., Sousa, P., Kedad, Z., & Salgado, A. C.: Summarizing ontology-based schemas in PDMS. In Data Engineering Workshops (ICDEW), pp. 239-244 (2010)
15. Hasan, R.: Generating and summarizing explanations for linked data, ESWC, pp. 473-487 (2014)
16. Donaway, R. L., Drummey, K. W., Mather, L.A.: A comparison of rankings produced by summarization evaluation measures, NAACL-ANLP Workshop, 4, pp. 69-78 (2000)
17. Maedche, A., Staab, S.: Measuring similarity between ontologies. Knowledge engineering and knowledge management: Ontologies and the semantic web, pp. 251-263 (2002)
18. Castano, S., De Antonellis, V., Fugini, M. G., Pernici, B.: Conceptual schema analysis: techniques and applications. TODS, 23(3), 286-333 (1998)
19. Liu, X., Tian, Y., He, Q., Lee, W.C., McPherson, J.: Distributed Graph Summarization. CIKM, pp. 799-808 (2014)
20. Khatchadourian, S., Consens, M.P.: ExpLOD: Summary-Based Exploration of Interlinking and RDF Usage in the Linked Open Data Cloud. ESWC, pp. 272-287 (2010)