Exploring RDFS KBs Using Summaries

Georgia Troullinou[1], Haridimos Kondylakis[1], Kostas Stefanidis[2], and Dimitris Plexousakis[1]

[1] ICS-FORTH, Heraklion, Greece, {troulin,kondylak,dp}@ics.forth.gr
[2] University of Tampere, Tampere, Finland, kostas.stefanidis@uta.fi

Abstract. Ontology summarization aspires to produce an abridged version of the original data source highlighting its most important concepts. However, in an ideal scenario, the user should not be limited only to static summaries. Starting from the summary, s/he should be able to further explore the data source requesting more detailed information for a particular part of it. In this paper, we present a new approach enabling the dynamic exploration of summaries through two novel operations *zoom* and *extend*. Extend focuses on a specific subgraph of the initial summary, whereas zoom on the whole graph, both providing granular information access to the end-user. We show that calculating these operators is NP-complete and provide approximations for their calculation. Then, we show that using extend, we can answer more queries focusing on specific nodes, whereas using global zoom, we can answer overall more queries. Finally, we show that the algorithms employed can efficiently approximate both operators.

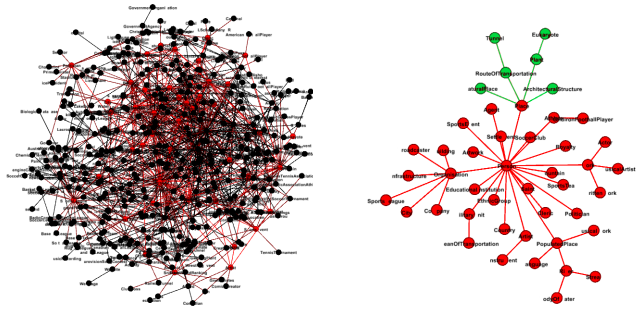
1 Introduction

The recent explosion of the Web of Data and the associated Linked Open Data (LOD) initiative have led to an enormous amount of widely available RDF datasets [6]. These datasets often have extremely complex schemas, which are difficult to comprehend, limiting the exploitation potential of the information they contain. As a result, there is an increasing need to develop methods and tools that facilitate the quick understanding and exploration of these data sources [9, 19].

To this direction, many approaches focus on generating ontology summaries [21, 29, 24, 25]. Ontology summarization [30] is defined as the process of distilling knowledge from an ontology in order to produce an abridged version. Although generating summaries is an active field of research, most of the works focus only on identifying the most important nodes, exploit limited semantic information or produce static summaries, limiting the exploration and the exploitation potential of the information they contain. In addition, although exploration operators over summaries have already been identified as really useful (e.g. [15]), the available approaches so far are limited, expanding only the hierarchy and the connections of selected nodes [11]. As a result, there is an increasing need to develop methods and tools in order to facilitate the understanding and exploration of various data sources, through exploration operators on summaries.

Consider for example that we would like to get a quick view of the DBpedia version 3.8 shown in Fig. 1(a). By visualizing the graph of the schema, it is difficult to understand the contents of the KB. Even if we highlight the most representative nodes (the red

ones), according to some importance measure (e.g. Betweenness) the problem persists. Now consider selecting the top-k most representative nodes and connecting them. The result is shown in Fig. 1(b). Here, we can better understand the contents of the DBpedia v3.8. However, still the user might find the presented information overwhelming and s/he would like to see less information, focusing only on the top-10 nodes. Ideally, s/he should be able to zoom-in and zoom-out at will in the presented graph to understand the contents at a selected granularity level. More than this, s/he might want to have more detailed information not only on the whole schema graph but on a selected subset of it. This could happen by selecting some nodes, requesting more details on those. Those details could be offered in terms of showing other nodes dependent on the selected ones as shown in Fig. 1(b) (green nodes). Although exploration operators over summaries have already been identified as useful (e.g. [15]), the available approaches are limited, expanding only the hierarchy and the connections of the selected nodes.



(a) (b)

Fig. 1: The DBpedia 3.8 schema graph (a) and a schema summary (b) generated using [17].

Motivated by the lack of an effective method to explore KBs starting from summaries, we have developed RDFDigest+. RDFDigest+ is a system that transparently and efficiently handles exploratory operations on large KBs. In its core, it employs an algebra where two operators are treated as first-class citizens in various exploration scenarios. Our algebra contains the *extend* and the *zoom* operators with particular semantics. Extend focuses on a specific subgraph of the initial summary, whereas zoom on the whole graph, both providing granular information access to the end-user.

More specifically, in this paper, we focus in RDFS ontologies and demonstrate an efficient and effective method to enable exploration of RDFS KBs, using schema summaries that can be extended and zoomed according to user selections. Our contributions are the following:

- We present RDFDigest+, a novel system that is able to generate summaries, enabling further exploration using zoom and extend operations.
- Summary generation is a two-steps process. First, all schema nodes are ranked according to various measures, and then, the top-k selected nodes are linked us-

ing edges that introduce the minimum number of additional nodes over the initial schema graph.
- Over these generated summaries, we enable zoom-in and zoom-out operations to get granular information, adding more important nodes or removing existing ones from the generated summary.
- In addition, through the extend operator, we allow selecting a subset of the presented nodes to visualize other dependent nodes.
- We provide algorithms for calculating the aforementioned operators on a given schema graph and we show that the problem is NP-complete. To this end, we provide effective and efficient approximations as well.
- We demonstrate the added value of these operators, evaluating summary's ability to answer the most-frequent real users queries, and we show that the approximate algorithms proposed can efficiently approximate both operators.

To our knowledge, this is the first approach that combines summaries with both zoom and extend operations, enabling effectively and efficiently the granular exploration of a KB. The rest of this paper is structured as follows: In Section 2, we present preliminaries and, in Section 3, we provide more details on schema summarization. Then, in Section 4, we introduce our ontology exploration operations. In Section 5, we present our experimental evaluation and, in Section 6, we discuss related work. Finally, in Section 7, we conclude this paper and present directions for further work.

2 Preliminaries

In this paper, we focus on RDFS KBs, as RDFS is among the widely-used standards for publishing and representing data on the Web. Our approach handles OWL ontologies as well, considering however only the RDFS part of these ontologies. The representation of knowledge in RDF is based on triples of the form (subject, predicate, object). RDF datasets have attached semantics through RDFS [1], a vocabulary description language. Representation of RDF data is based on three disjoint and infinite sets of *resources*, namely: URIs ($\mathcal{U}$), literals ($\mathcal{L}$) and blank nodes ($\mathcal{B}$). We impose typing on resources, so we consider three disjoint sets of resources: classes ($\mathbf{C} \subseteq \mathcal{U} \cup \mathcal{B}$), properties ($\mathbf{P} \subseteq \mathcal{U}$), and individuals ($\mathbf{I} \subseteq \mathcal{U} \cup \mathcal{B}$). The set $\mathbf{C}$ includes all classes, including RDFS classes and XML datatypes (e.g., xsd:string, xsd:integer). The set $\mathbf{P}$ includes all properties, except rdf:type, which connects individuals with the classes they are instantiated under. The set $\mathbf{I}$ includes all individuals, but not literals. In addition, our approach adopts the unique name assumption, i.e. resources identified by different URIs are different.

Here, we will follow an approach similar to [26], which imposes a convenient graph-theoretic view of RDF data that is closer to the way the users perceive their datasets. As such, we separate between the schema and the instances of an RDFS KB, represented in separate graphs (G_S and G_I, respectively). The schema graph contains all classes and the properties the classes associated with (via the properties domain/range specification); multiple domains/ranges per property are allowed, by having the property URI be a label on the edge, via a labeling function λ, rather than the edge itself. The instance graph contains all individuals, and the instantiations of schema properties; the labeling

function λ applies here as well for the same reasons. Finally, the two graphs are related via the τ_c function, which determines the class(es) each individual is instantiated under.

Definition 1. ***(RDFS KB)*** *An RDFS KB is a tuple $V = \langle G_S, G_I, \lambda, \tau_c \rangle$, where:*

- *G_S is a labelled directed graph $G_S = (V_S, E_S)$ such that V_S, E_S are the nodes and edges of G_S, respectively, and $V_S \subseteq \mathbf{C} \cup \mathcal{L}$.*
- *G_I is a labelled directed graph $G_I = (V_I, E_I)$ such that V_I, E_I are the nodes and edges of G_I, respectively, and $V_I \subseteq \mathbf{I} \cup \mathcal{L}$.*
- *A labelling function $\lambda : E_S \cup E_I \mapsto 2^{\mathbf{P}}$ determines the property URI that each edge corresponds to (properties with multiple domains/ranges may appear in more than one edge).*
- *A function $\tau_c : \mathbf{I} \mapsto 2^{\mathbf{C}}$ associating each individual with the classes that it is instantiated under.*

In the following, we will write $p(v_1, v_2)$ to denote an edge e in G_S, where $v_1, v_2 \in V_S$, or G_I, where $v_1, v_2 \in V_I$, from node v_1 to node v_2, such that, $\lambda(e) = p$. In addition, for brevity, we will call schema node a node $s \in V_S$, class node a node $c \in \mathbf{C} \cap V_S$, and instance node a node $i \in \mathbf{I} \cap V_I$. A path from a node v_s to v_i, denoted by $path(v_s \rightarrow v_i)$, is the finite sequence of edges, which connect a sequence of nodes, starting from v_s and ending at v_i. The length of a path, denoted by $dpath(v_s \rightarrow v_i)$, is the number of the edges that exist in that path. Finally, having a schema graph G_S, the closure of G_S, denoted by $Cl(G_S)$, contains all triples that can be inferred from G_S using inference. From now on, when we use G_S, we will mean $Cl(G_S)$ for reasons of simplicity, unless stated otherwise. This is to ensure that the result will be the same, independent of the number of inferences applied on an input schema graph G_S.

3 Schema Summarization

Schema summarization aims to highlight the most representative concepts of a schema, preserving important information and reducing the size and the complexity of the whole schema. Central questions to summarization are (i) how to rank the schema nodes according to an importance measure, and (ii) how to link the top-k ones in order to produce a valid sub-schema graph.

3.1 Identifying Important Nodes in RDFDigest+

To identify the most important nodes, RDFDigest+ employs a variety of centrality measures like Degree, Bridging Centrality, Harmonic Centrality, Radiality, Ego Centrality and Betweenness [17]. As [17] shows, among these measures, Betweenness produces summaries with a better quality. In addition, in this paper we explore for the first time to this purpose, PageRank and HITS, two additional well-known centrality measures [5]. Specifically, the importance measures (IM) we are going to explore for our experiments, for selecting the top-k most important nodes are the following:

- *Betweenness (BE).* The number of the shortest paths from all nodes to all others that pass through a node.

- *PageRank (PR)*. This centrality measure assigns a score based on node's connections, and their connections connections. PageRank takes link direction and weight into account so links can only pass influence in one direction, and pass different amounts of influence.
- *HITS (HT)*. HITS algorithm is based on the idea that in the Web, and in all document collections which can be represented by directed networks, there are two types of important nodes: hubs and authorities. Hubs are nodes which point to many nodes of the type considered important. Authorities are these important nodes.

Independently of the importance measure (IM) selected, since those measures have been developed for generic graphs, we adapt them to be used for RDFS graphs. To achieve that we first normalize each measure IM on a scale of 0 to 1:

$$normal(IM(v)) = \frac{IM(v) - \min(IM(G_S))}{\max(IM(G_S)) - \min(IM(G_S))} \quad (1)$$

where $IM(v)$ is the importance value of a node v in G_S, and $min(IM(G_S))$ is the minimum and $max(IM(G_S))$ is the maximum importance value in G_S.

Similarly, we normalize the number of instances (InstV) that belong to a schema node. As such, the *adapted importance measure* (AIM) of each node is the sum of the normalized values of the importance measures and the instances.

$$AIM(v) = normal(IM(v)) + normal(InstV(v)) \quad (2)$$

Next, let $TOP_k^{AIM}(V)$ be the function that returns the top-k nodes of an RDFS KB V, according to the selected adapted importance measure (AIM) - for brevity we will use $TOP_k(V)$ independently of the importance measure selected.

Overall, our system is flexible enough to enable the uninterrupted addition of new importance measures by adding new function calls. The diverse set of importance measures offered, enable exploring RDFS KBs according to the way users perceive importance, offering many alternatives and enhancing the exploration abilities of our system.

3.2 Linking Important Nodes

Having a way to rank the schema nodes of an RDFS KB according to the perceived importance, we then focus on selecting the paths that link those nodes, aiming to produce a valid sub-schema graph. As the main problem of previous approaches [17, 26] was the introduction of many additional nodes (besides the top-k ones), in this paper, we focus on selecting the paths that introduce the minimum number of additional nodes to the final summary graph. As such, we model the problem of linking the most important nodes as a variation of the well-known *Graph Steiner-Tree problem (GSTP)* [27]. The corresponding algorithm targets at minimizing the additional nodes introduced for connecting the top-k most important nodes [17]. However, the problem is NP-hard, and as such approximation algorithms should be used for large datasets.

3.3 Summary Schema Graph

Having identified ways for locating important nodes and, in turn, for connecting them, we define next the summary schema graph as follows:

Definition 2 (Summary Schema Graph of size n). *Let $V = \langle G_S, G_I, \lambda, \tau_c \rangle$ be an RDFS KB. A summary schema graph of size n for V is a connected schema graph $G'_S = (V'_S, E'_S)$, $G'_S \subseteq Cl(G_S)$, with:*

- $V'_S = TOP_k(V) \cup V_{ADD}$,
- $\forall v_i, v_j \in TOP_k(V)$, $\exists path(v_i \rightarrow v_j) \in G'_S$,
- V_{ADD} *represents the nodes in the summary used only to link the nodes in* $TOP_k(V)$,
- $\nexists$ *summary schema graph* $G''_S = (V''_S, E''_S)$ *of size* n *for* V*, such that,* $|V''_S| < |V'_S|$.

4 Exploration through Summaries

Getting the summaries, users can better understand the contents of a KB. However, still the user might find the presented information overwhelming and he/she may like to see less information, focusing for example, only on the top-10 nodes (zoom) or requesting more detailed information for a specific subgraph of the summary (extend).

4.1 The Extend Operator

The extend operator gets as input a subgraph of the schema graph and identifies other nodes that are depending on the selected nodes. Dependence has not only to do with distance, but with additional parameters, including importance. Like TF-IDF, the basic hypothesis here is that the greater the influence of a property on identifying a corresponding instance is, the less times it is repeated, or in other words, infrequent properties are more informative than frequent ones. This way, we define the dependence between two classes as a combination of their cardinality closeness (defined in the sequel), the adapted importance measures (AIM) of the classes and the number of edges appearing in the path connecting these two classes. So, dependence is defined as:

$$Dependence(u, v) = \frac{AIM(u) - \sum_{i \in Y} \frac{AIM(i)}{CC((i-1),i)}}{dpath(u \rightarrow v)} \tag{3}$$

where the cardinality closeness CC is defined for a pair of classes as the number of distinct edges over the number of all edges between them. Formally:

Definition 3 (Cardinality Closeness). *Let c_k, c_s be two adjacent schema nodes and $u_i, u_j \in G_I$ such that $\tau_c(u_i) = c_k$ and $\tau_c(u_j) = c_s$. The cardinality closeness of $p(c_k, c_s)$, namely the $CC(p(c_k, c_s))$, is defined as:*

$$CC(p(c_k, c_s)) = \frac{1 + |c|}{|c|} + \frac{DistinctV(p(u_i, u_j))}{Instances(p(u_i, u_j))} \tag{4}$$

where $|c|, c \in C \cap V_S$*, is the number of nodes in the schema graph,* $DistinctV(p(u_i, u_j))$ *is the number of distinct* $p(u_i, u_j)$ *and* $Instances(p(u_i, u_j))$ *is the number of* $p(u_i, u_j)$*. When there are no instances,* $Instances(p(u_i, u_j)) = 1$ *and* $DistinctV(p(u_i, u_j)) = 0$*.*

As we move away from a node, the dependence becomes smaller by calculating the differences of AIM across a selected path in the graph. We penalize additionally dependence dividing by the distance of the two nodes. The highest the dependence of a path, the more appropriate is the first node to represent the final node of the path. Also note that $Dependence(u, v)$ is different than $Dependence(v, u)$, since the dependence of a more important node towards a less important node is higher than the other way around, although, they share the same cardinality closeness. To identify the dependent nodes of a selected node, we use the function $dependend(u_i, range, number\_of\_nodes)$ that returns at most $number\_of\_nodes$ nodes depending on u_i with a distance at most $range$.

The *extend* operator takes into account a particular subgraph of a summary schema graph, and is defined as follows:

Definition 4 (Extend operator). *Let* $G'_S = (V'_S, E'_S)$ *be the summary schema graph of an RDFS KB* $V = \langle G_S, G_I, \lambda, \tau_c \rangle$*. The extend operator, i.e.,* $extend(G_e)$*, takes as input a subgraph* $G_e = (V_e, E_e)$ *of* G'_S*,* $G_e \subseteq G'_S$*, and returns a connected schema graph* $G'_e = (V'_e, E'_e)$*,* $V_e \subseteq V'_e$*, for which:*

- $G'_e \subseteq Cl(G_S)$*,*
- $V'_e \backslash V_e = V_d \cup V_{ADD'}$*, where* V_d *includes,* $\forall v_i \in V_e$*, all nodes* v_j*, such that,* $dependend(v_j, range, number\_of\_nodes) = v_i$*, and* $V_{ADD'}$ *the nodes that link the nodes in* V_d *with the other summary nodes,*
- $\forall v_i \in V_d \cup TOP_k(V)$*,* $\exists path(v_x \rightarrow v_y) \in G'_e$*,*
- $\nexists\, G''_e = extend(G_e) = (V''_e, E''_e)$*, such that,* $|V''_e| < |V'_e|$*.*

Algorithm 1 presents the extend algorithm. The algorithm identifies the dependent nodes (lines 2-5) using the depencence function. Due to lack of space, the detailed description of the algorithm used for locating the $dependent$ nodes is omitted, however abstractly, it starts from u_i and calculate the dependence of the adjacent nodes expanding progressively the range until it reaches the $number\_of\_nodes$. Next, the algorithm tries to link the top-k nodes using the Steiner-Tree algorithm (line 6). However, as the Steiner-Tree algorithm is NP-complete, our problem is NP-complete as well.

Two optimizations that we explore in this work are the following:

CHINS. CHINS is an approximation of the Steiner-Tree algorithm [27] proved to have a worst case bound of 2, i.e., $Z_T / Z_{opt} \leq 2 \cdot (1 - l/|Q|)$, where Z_T and Z_{opt} denote the objective function values of a feasible solution and an optimal solution respectively, Q the set of nodes to be linked (for the extend operator the top-k nodes and the selected dependent ones) and l a constant [3]. The algorithm proceeds as follows:

1. Start with a partial solution consisting of a single selected node.
2. While the solution does not contain all selected nodes do
   find the nearest nodes $u* \in V_t$ and $p*$ being a top-k node not in V_t.

Algorithm 1 Extend

Input $G'_S = (V'_S, E'_S)$ the summary schema graph of G_S, $G_e = (V_e, E_e)$ the selected summary schema subgraph

Output $G'_e = (V'_e, E'_e)$ the result schema graph

1: **procedure** EXTEND
2: $V'_e = V'_S$
3: **for** each v_i in V_e **do**
4: $V'_e = V'_e \cup dependent(v_i, range, number\_of\_nodes)$
5: **end for**
6: Calculate E'_e using the Steiner-Tree algorithm over G_S with the nodes in V_e as terminals
7: **end procedure**

As such, for each node to be linked, the algorithm has to visit at worst the whole set of nodes and edges of the graph, and the corresponding complexity is $O(Q \cdot |V + E|)$. CHINS has been proved to offer an optimal trade-off between quality of the generated summaries and execution time [17], when used for generating summaries.

Shortest Paths. CHINS starts from a single node extending one by one the set of selected nodes. However, having the nodes in the summary already, there is no need to start from the first node. As such, another approximation could be to start with the nodes already available in the summary and then proceed to step 2 of CHINS. The algorithm for each one of the $|Q \backslash TOP_K(V)|$ nodes needs at worst to visit the whole graph. This way, the worst-case complexity of the algorithm is $O(|Q \backslash TOP_K(V)| \cdot |V + E|)$.

Dependent paths. In order to calculate the dependence between the selected nodes and the ones introduced by the $dependent$ functions, the visited paths can be recorded and use these, already visited paths for connecting the selected nodes with the original summary. So, in this approximation, instead of finding the shortest path between the existing summary and each dependent node, we calculate the shortest path between the extended and the dependent node, which is already calculated in the previous step (the *dependent* function). The complexity remains the same with the previous algorithm ($O(|Q \backslash TOP_K(V)| \cdot |V + E|)$), since only the $|Q \backslash TOP_K(V)|$ nodes are considered sequentially for linking them to the existing summary.

4.2 The Zoom Operator

In this section, we focus on zooming operations, by exploiting the schema graph as a whole. That is, we introduce the *zoom-out* and *zoom-in* operators to produce more detailed or coarse summary schema graphs. To this end, we consider the n' schema nodes with the highest importance in G_S, where n' can be either greater than n, for achieving a *zoom-out*, or smaller than n, for achieving a *zoom-in*, where n represents the number of the most important nodes in a given summary.

Definition 5 (Zoom-out operator). *Let $G'_S = (V'_S, E'_S)$ be the summary schema graph of size n of an RDFS KB $V = \langle G_S, G_I, \lambda, \tau_c \rangle$. The zoom-out operator $zoom_{out}(G'_S, n')$, with $n' > n$, returns a connected schema graph $G'_{zo} = (V'_{zo}, E'_{zo})$, for which:*

- $G'_{zo} \subseteq Cl(G_S)$,
- $V'_{zo} = V'_S \cup TOP \cup V_{ADD}$, *where* $TOP = TOP_{n'}(V) \backslash V'_S$,
- $\forall v_i \in TOP$, $\exists v_j \in V'_S$, *such that,* $\exists path(v_i \rightarrow v_j) \in G'_{zo}$,
- V_{ADD} *represents the nodes in* G'_{zo} *used only to link the nodes in TOP,*
- $\nexists\, G''_z o = zoom_{out}(G'_S, n') = (V''_{zo}, E''_{zo})$, *such that,* $|V''_z o| < |V'_z o|$.

Definition 6 (Zoom-in operator). *Let* $G'_S = (V'_S, E'_S)$ *be the summary schema graph of size* n *of an RDFS KB* $V = \langle G_S, G_I, \lambda, \tau_c \rangle$. *The zoom-in operator* $zoom_{in}(G'_S, n')$, *with* $n' < n$, *returns a connected schema graph* $G'_{zi} = (V'_{zi}, E'_{zi})$, *for which:*

- $G'_{zi} \subseteq G'_S$,
- $V'_{zi} = TOP_{n'}(V) \cup V_{ADD}$,
- V_{ADD} *represents the nodes in* G'_{zi} *used only to link the nodes in* $TOP_{n'}(V)$,
- $\nexists\, G''_{zi} = zoom_{in}(G'_S, n') = (V''_{zi}, E''_{zi})$, *such that,* $|V''_{zi}| < |V'_{zi}|$.

The simplest approach for zooming-in/out, is to calculate from scratch the $TOP_{n'}(V)$ and then to use the Steiner-Tree algorithm from scratch to link the selected nodes. However, since we already have an existing summary as a basis for our zoom operations, we explore the following approximations.

Zoom-in. Remove the nodes in $TOP_n(V) \backslash TOP_{n'}(V)$ and their connections without recalculating the Steiner-Tree algorithm for $TOP_{n'}(V)$ – this might leave additional nodes in the resulting summary.

Zoom-out - CHINS. Add the nodes in $TOP_{n'}(V) \backslash TOP_n(V)$ and link them with the existing summary, using the CHINS approximation algorithm.

Zoom-out - Shortest Paths. Add the nodes in $TOP_{n'}(V) \backslash TOP_n(V)$ and link them with the existing summary, using the *Shortest Paths* approximation algorithm.

5 Evaluation & Implementation

To evaluate our approach, we use the version 3.8 of DBpedia[3], which is consisted of 359 classes, 1323 properties and more that 2.3M instances, and offers an interesting use-case for exploration. To identify the quality of our approach, we use a query log containing 50K user queries provided by the DBpedia SPARQL end-point for the corresponding DBpedia version. Our goal is to assess the percentage of the queries that can be answered solely by using the generated schema summary along with the corresponding instances, i.e. the *coverage* of the queries from a schema summary.

Having a summary, we can calculate for each query the percentage of the classes and properties that are included in the summary. A class/property appears within a query either directly or indirectly. Directly when the said class/property appears within a triple pattern of the query. Indirectly for a class is when the said class is the type of an instance or the domain/range of a property that appear in a triple pattern of the query. Indirectly for a property is when the said property is the type of an instance. Having the percentages of the classes and properties included in the summary, the query coverage is the weighted sum of these percentages. As our summaries are node-based (they are generated based on the top-k most important nodes; in zoom we add/remove important nodes; in extend we add the dependent nodes) the weight on the nodes is larger than the one on the properties (for our experiments we used 0.8 for nodes and 0.2 for edges).

[3] `http://wiki.dbpedia.org/`

5.1 Quality - Evaluating the Zoom Operator

In this section, we evaluate the quality of the zoom-out operator. To do that we start from a summary containing 10% of the initial schema graph, and we zoom-out progressively by 10%, until we reach the 40% of the schema graph. Having the *coverage* of each query, we can calculate the average coverage for all queries in our log. In essence, an average coverage of 70% means that on average the 70% of the queries in the query log can be answered only using the summary accompanied with its corresponding instances. As when zooming-out, the next more important nodes are added to the summary, we expect that the average coverage of all queries should grow accordingly. The results are shown in Fig. 2, whereas the actual improvement is shown in Fig. 3. As

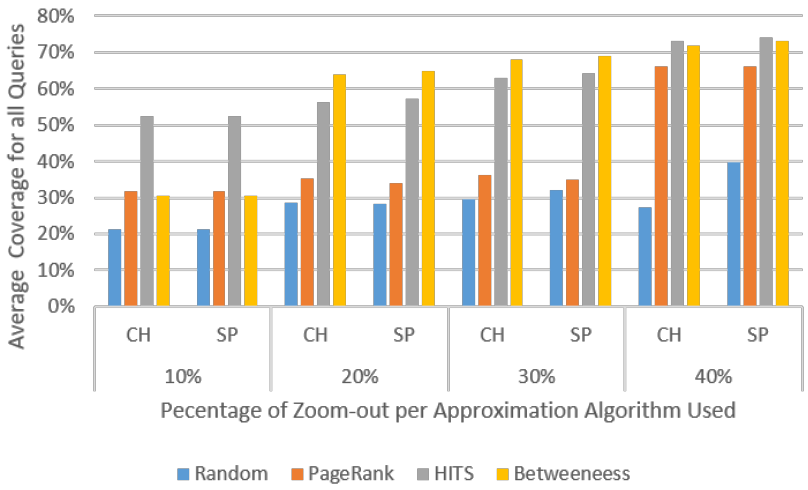


Fig. 2: Zooming-out using various centrality measures and approximation algorithms CHINS (CH) and Shortest Paths (SP).

we can observe, indeed as the percentage of the summary increases, more queries are covered by the result summary. In addition, HITS and Betweenness perform better, competing each other in all cases. Specifically, HITS presents a more stable behavior with the best coverage from the smallest zoom-out percentage, while Betweenness performs better from the 20% zoom-out and on. PageRank is always worse than HITS and Betweenness. As a baseline we added the Random bar as well, where we randomly select nodes from the schema graph (connecting them with the corresponding measure). Even if some-times randomly adding more nodes improves a bit the results, overall, this is the approach with the worst performance, clearly showing the benefits of our approach. Regarding the actual improvement, we observe that CHINS and Shortest Paths return results of the same quality, with Shortest Paths being slightly better in some cases. In this sense, Betweenness appears to be the most stable measure with improvements around 35% to 45%, while PageRank shows a good improvement, around 35%,

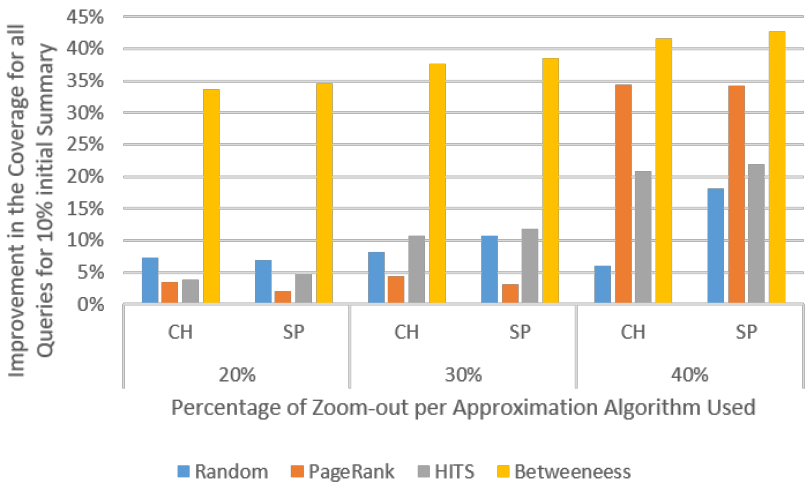


Fig. 3: Improvement on Zooming-out using various centrality measures and approximation algorithms CHINS (CH) and Shortest Paths (SP).

for cases in which a 40% zoom-out is performed. Due to space limitations, we omit the results of the zoom-in operator that presents similar behavior.

5.2 Quality - Evaluating the Extend Operator

Next, we evaluate the extend operator. To do that, we start again from a summary containing 10% of the initial schema graph, and we extend progressively requesting to extend 10% of the available nodes in the summary, until we reach 40% of the initial summary schema graph being extended.

As now we are interested in getting information relevant to particular selected nodes, and not for the whole schema graph, we calculate the average coverage for the queries including only classes from the selected part to be extended. In this case, an average coverage of 70% means that on average the 70% of the queries in the query log, including one of the extended nodes, can be answered only using the summary accompanied by its corresponding instances. As when more nodes related to the extended ones, are added to the summary, we expect that the average coverage of those queries should grow accordingly. The results are shown in Fig. 4, whereas the actual improvement is shown in detail in Fig. 5.

Overall, we observe here that indeed the more nodes we extend, the more "local" queries are covered. In addition, the Shortest Paths algorithm provides the best results in all cases, followed by CHINS. This is reasonable since the Shortest Paths algorithm targets at identifying the shortest path between the dependent nodes and the available summary, and as such, it prioritizes nodes closest to the ones to be extended. On the other hand, the Dependent paths algorithm does a minimum effort trying to connect the dependent nodes to the existing summary and this has a direct effect on the quality of

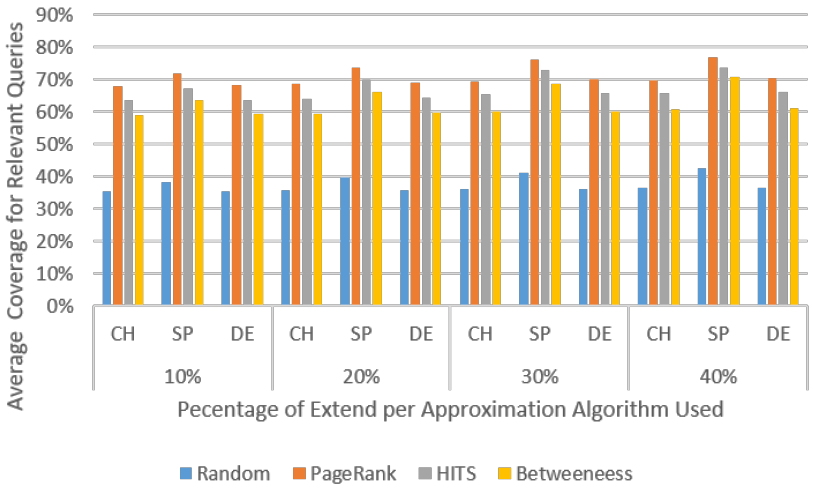


Fig. 4: Extend using HITS and Betweenness, and the approximation algorithms random (RA), CHINS (CH), Shortest Paths (SP) and Dependent (DE).

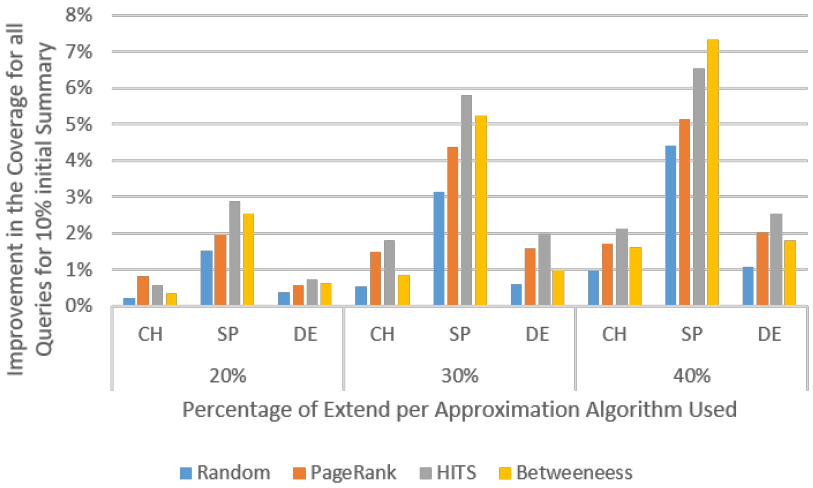


Fig. 5: Improvement on extending using HITS and Betweeneess, and the approximation algorithms random (RA), CHINS (CH), Shortest Paths (SP) and Dependent (DE).

the produced summary. PageRank presents the best coverage, on average around 68% to 78%, while HITS follows with coverage around 65% to 73%. In turn, Betweenness has a coverage around 59% to 72%, while, as expected, Random presents the worst behavior with coverage from 35% to 40%. Overall, even if PageRank has the best performance, we observe that Betweeness has the best improvement.

5.3 The RDFDigest+ System

All aforementioned measures and algorithms are available online on the RDFDigest+ system[4], a novel system that enables effective and efficient RDFS KB exploration using summaries. An instance of RDFDigest+ is shown in Figure 6.

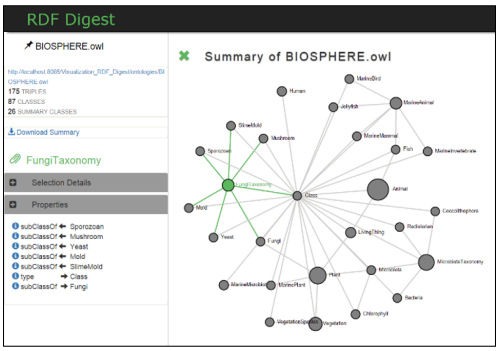


Fig. 6: The RDFDigest+ System.

Users can upload their own datasets, and RDFDigest+ produces a visual summary identifying and linking the most important nodes in the KB. In the presented summary graph, the size of a node depends on its importance. By clicking on a node, additional metadata (e.g. the number of instances, and the connected properties and instances) are provided to enhance the ontology understanding. Further exploration of the data source is allowed by clicking on the details (on the left) of the selected class and properties. When clicked, its instances and connections appear in a pop-up window. In addition, exploration of the data source is allowed by double-clicking on a node to extend the summary on that specific node. Besides a specific node, a whole area can be selected, requesting more detailed information to be presented regarding the selected nodes. The summary can be zoomed-in and zoomed-out in order to present more detailed or more generic information regarding the whole summary. Finally, the user is able to download the summary as a valid RDFS document.

[4] `http://rdfdigest.ics.forth.gr`

6 Related Work

According to [20], an effective ontology exploration system should provide a number of core functionalities, such as providing a high level overview of the data, zooming in specific parts of the data and filtering out irrelevant parts.

Ontology Visualization Systems. Towards this direction, toolkits like Protege [16], TopBraid Composer [2] and Neon [8], include visualization plug-ins using the node-link diagram paradigm to represent entities in an ontology and their taxonomy to domain relationships. In addition, many plug-ins, like OwlViz in Protege and Graph View in TopBraid, allow navigating the ontology hierarchy by expanding and hiding nodes.

SpaceTree [18] follows the node-link paradigm as well, but is able to maximize the nodes on display by assessing the available display space. It also avoids clutter by utilizing informative preview icons giving the user an idea of the size and shape of the corresponding subtrees. CropCircles [28] on the other hand, uses geometric containment as an alternative to classing node-link displays sacrificing space to make it easier for users to understand the topological relations in an ontology. Hybrid solutions, like Jambalaya [23] and Knoocks [12], combine containment-based and node-link approaches by providing alternative integrated views of the two paradigms, whereas other approaches, like [7], are based on the notion of distorting the view of the presented graph to combine context and focus. The node on focus is usually the central one and the rest of the nodes are presented around it, reduced in size until they reach a point that they are no longer visible. Finally, WebVOWL [14] implements the Visual Notation for OWL Ontologies (VOWL) by providing graphical depictions for elements of the Web Ontology Language (OWL) that are combined to a force-directed graph layout representing the ontology.

However, all aforementioned approaches in essence, use geometric techniques to provide the necessary abstraction, such as hyperbolic or force-directed graphs, geometric containment or miniature sub-trees. However, we argue that an ideal visualization approach should start with the most important elements of the ontology allowing then progressively the users to explore other less important areas.

Ontology Summarization Systems. Besides pure ontology visualization systems, ontology summarization systems have adopted as well zooming functionalities. An example is KC-Viz [15], which focuses on the key concepts of the ontology based on psycholinguistic criteria. Our system on the other hand, allows users to select multiple measures for identifying importance. KC-Viz provides a set of navigation and visualization mechanisms, including flexible zooming into and hiding of specific parts of an ontology. However, this work is limited in selectively expanding the hierarchy and the connections of selected nodes, whereas in our case besides zooming, we also visualize dependent nodes enabling further exploration of the data source.

[13] supports zoom, filter, details-on-demand, relate, history and extract operations using hierarchical connected circles to provide overview, indented trees to relate different concepts and node-links for filtering and details on-demand, enabling the users to choose the level of semantic zoom. However, the operations performed are not formalized, the corresponding algorithms are not presented and an evaluation is completely missing from the aforementioned work.

[10] proposes a tool that supports three visual exploration options. The first one, named *landmark view*, provides an overview of the class(property) taxonomy giving only representative classes in the hierarchy - selected automatically by a set of statistics measures and user preferences. Then, a user can further explore a specific area by extending(or collapsing) branches. The *local view* displays the full hierarchy of a set of classes (properties) whereas *the axiom view*, provides information about a selected class and its connectivity in the ontology. Compared to our work, this approach is limited mostly on hierarchical structures.

7 Conclusions

In this paper[5] we present a novel platform enabling KB exploration operations over summaries. We introduce the zoom and extend operations, focusing on the number of important nodes of the generated summary, and on getting more detailed information for selected schema summary nodes, respectively. We explore various approximation algorithms showing that we can calculate efficiently the aforementioned operations without sacrificing the quality of the result summary. In fact, we show that the Shortest Paths algorithm provides an optimal trade-off between efficiency and quality.

To the best of our knowledge RDFDigest+ is currently the only system enabling such exploration operations over summaries. As future work, we intent to enable KB exploration at the instance level as well, going from schema summaries to instance summaries, enabling zoom and extend operations both as schema and instance level, or exploiting big data frameworks to speed the summarization process [4]. Moreover, given the dynamically evolving datasets we handle, users are often interested in the state of affairs on previous versions of the datasets, along with their corresponding summaries. To address this need, archiving policies [22] typically store adequate deltas between versions, which are generally small, but this would create the overhead of generating versions at query time. As a direct extension of our system, we will study the trade-offs involved when focusing on archiving dynamic RDF summaries.

References

1. *RDF Schema 1.1*. Available online: http://www.w3.org/TR/rdf-schema/, (last accessed April 2018).
2. *TopBraid Composer*. Available online: https://www.topquadrant.com/tools/ide-topbraid-composer-maestro-edition/ , (last accessed October 2017).
3. In D.-Z. Du, J. M. Smith, and J. H. Rubinstein, editors, *Advances in Steiner Trees*. Kluwer Academic Publishers, 2000.
4. G. Agathangelos, G. Troullinou, H. Kondylakis, K. Stefanidis, and D. Plexousakis. RDF query answering using apache spark: Review and assessment. In *ICDE*, 2018.
5. P. Boldi and S. Vigna. Axioms for centrality. *Internet Mathematics*, 10(3-4):222–262, 2014.
6. V. Christophides, V. Efthymiou, and K. Stefanidis. *Entity Resolution in the Web of Data*. Synthesis Lectures on the Semantic Web: Theory and Technology. Morgan & Claypool Publishers, 2015.

[5] The work was partially supported by the TEKES Finnish project Virpa D. and by the iManageCancer EU project (H2020, #643529).

7. K. X. S. de Souza, A. D. dos Santos, and S. R. M. Evangelista. Visualization of ontologies through hypertrees. In *CLIHC*, 2003.
8. M. Erdmann and W. Waterfeld. Overview of the neon toolkit. In *Ontology Engineering in a Networked World.*, pages 281–301. 2012.
9. P. Fafalios, V. Iosifidis, K. Stefanidis, and E. Ntoutsi. Multi-aspect entity-centric analysis of big social media archives. In *TPDL*, 2017.
10. Z. L. Jiao, Q. Liu, Y. Li, K. Marriott, and M. Wybrow. Visualization of large ontologies with landmarks. In *GRAPP & IVAPP*, 2013.
11. H. Kondylakis, G. Troullinou, K. Stefanidis, and D. Plexousakis. Beyond summaries for ontology exploration. *ERCIM News*, 2018(113), 2018.
12. S. Kriglstein and G. Wallner. Knoocks - A visualization approach for OWL lite ontologies. In *CISIS*, 2010.
13. S. Kuhar and V. Podgorelec. Ontology visualization for domain experts: A new solution. In *International Conference on Information Visualisation, IV*, 2012.
14. S. Lohmann, V. Link, E. Marbach, and S. Negru. Webvowl: Web-based visualization of ontologies. In *EKAW 2014 Satellite Events*, 2014.
15. E. Motta, S. Peroni, N. Li, and M. d'Aquin. Kc-viz: A novel approach to visualizing and navigating ontologies. In *EKAW*, 2010.
16. M. A. Musen. The protégé project: a look back and a look forward. *AI Matters*, 1(4):4–12, 2015.
17. A. Pappas, G. Troullinou, G. Roussakis, H. Kondylakis, and D. Plexousakis. Exploring importance measures for summarizing RDF/S KBs. In *ESWC*, 2017.
18. C. Plaisant, J. Grosjean, and B. B. Bederson. Spacetree: Supporting exploration in large node link tree, design evolution and empirical evaluation. In *InfoVis*, 2002.
19. Y. Roussakis, I. Chrysakis, K. Stefanidis, G. Flouris, and Y. Stavrakas. A flexible framework for understanding the dynamics of evolving RDF datasets. In *ISWC*, 2015.
20. B. Shneiderman. The eyes have it: A task by data type taxonomy for information visualizations. In *IEEE Symposium on Visual Languages*, 1996.
21. P. Silvio, M. Enrico, and d'Aquin Mathieu. Identifying key concepts in an ontology, through the integration of cognitive principles with statistical and topological measures. In *ASWC*, 2008.
22. K. Stefanidis, I. Chrysakis, and G. Flouris. On designing archiving policies for evolving RDF datasets on the web. In *ER*, 2014.
23. M. D. Storey, N. F. Noy, M. A. Musen, C. Best, R. W. Fergerson, and N. A. Ernst. Jambalaya: an interactive environment for exploring ontologies. In *IUI*, 2002.
24. G. Troullinou, H. Kondylakis, E. Daskalaki, and D. Plexousakis. RDF digest: Efficient summarization of RDF/S kbs. In *ESWC*, 2015.
25. G. Troullinou, H. Kondylakis, E. Daskalaki, and D. Plexousakis. RDF digest: Ontology exploration using summaries. In *ISWC*, 2015.
26. G. Troullinou, H. Kondylakis, E. Daskalaki, and D. Plexousakis. Ontology understanding without tears: The summarization approach. *Semantic Web*, 8(6):797–815, 2017.
27. S. Voß. Steiner's problem in graphs: Heuristic methods. *Discrete Applied Mathematics*, 40(1):45–72, 1992.
28. T. D. Wang and B. Parsia. Cropcircles: Topology sensitive visualization of OWL class hierarchies. In *ISWC*, 2006.
29. G. Wu, J. Li, L. Feng, and K. Wang. Identifying potentially important concepts and relations in an ontology. In *ISWC*, 2008.
30. X. Zhang, G. Cheng, and Y. Qu. Ontology summarization based on rdf sentence graph. In *WWW*, 2007.