RDFDigest+: A Summary-driven System for KBs Exploration

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

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

Abstract. In this paper, we present RDFDigest+, a novel tool that enables effective and efficient RDF/S Knowledge Base (KB) exploration using summaries. The tool employs a diverse set of algorithms for identifying the most important nodes, offering a wide range of possibilities to capture importance. The selected nodes can be combined using multiple state of the art algorithms to generate a complete schema summary graph. In addition, 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.

1 Introduction

The recent explosion of the Web data and the associated Linked Open Data initiative have led to an enormous amount of widely available RDF datasets [1]. 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 [3].

In our work, we design and develop RDFDigest+(`http://rdfdigest.ics.forth.gr`) that focuses on three aspects: a) how to identify the most important nodes of an RDF/S KB, b) how to link those nodes in order to produce a valid sub-schema graph and c) how to enable the active exploration of the KB, presenting various statistical information and enabling zooming operators. Multiple importance measures [2] have been adapted from graph theory, offering a wide range of alternatives for identifying the importance of a node. To locate the proper paths connecting those nodes, we model the problem either as a graph Steiner-tree problem or as a maximum cost-spanning tree one [4]. Finally, over these generated summaries, we enable zoom-in and zoom-out operations, in order to get granular information, by 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 in order to visualize other dependent nodes. Both operators can be progressively applied to make the whole process more efficient (more details in [5]).

2 Approach

In this work, we separate between the schema and instances of an RDF/S KB, represented in separate graphs, G_S and G_I. The schema graph contains all classes and the properties they are associated with. The instance graph contains all individuals, and the instantiations of schema properties.

Identification of the most important nodes. For ranking the available schema nodes based on their importance, we use the following measures: Degree, Betweeness, Bridging Centrality, Harmonic Centrality, Radiality, Betweenness, PageRank and HITS [2] [5]. As the aforementioned measures have been developed for generic graphs, we adapt them to be used for RDF/S graphs. To achieve that first we normalize each importance measure and then the number of instances 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. Overall, our platform is flexible enough to enable the addition of new measures by just adding new function calls.

Linking most important nodes Having a way to rank the schema nodes of an RDF/S KB according to the perceived importance, we then select the top-k ones and focus on the paths that link those nodes, aiming to produce a valid sub-schema graph. RDFDigest+ offers two methods for achieving this. The first one focuses on identifying a *maximum cost spanning tree (MST)* in the graph, and on linking the selected nodes using paths from the selected tree. The corresponding algorithm is computationally efficient. However, although MST identifies the paths with the maximum weight in the whole graph, the paths selected out of the MST might not maximize overall the weight of the selected summary. Moreover, many additional nodes might be introduced in the result, since there is only one path to be selected between two nodes and in this path many other not important nodes might appear as well. Alternatively, we model the problem of linking the most important nodes as a variation of the well-known *Graph Steiner-Tree problem* where the goal is to minimize the additional nodes introduced for connecting the top-k nodes. However, the problem is NP-hard, and as such approximation algorithms should be explored.

Exploration through summaries. With summaries, users can better understand the KB contents. However, they might find the presented information overwhelming and may want to see complementary information.

Extend: The extend operator gets as input a subgraph of the summary schema graph and identifies other nodes that depend on the selected nodes. Dependence has not only to do with distance. Like TF-IDF, the basic hypothesis is that the greater the influence of a property on identifying a corresponding instance is, the less times it is repeated. Specifically, we define the dependence between two classes as a combination of their cardinality closeness CC, the AIM of the classes, and the number of nodes appearing in the path Y connecting these two classes. That is: $Dependence(u,v) = \frac{AIM(v) - \sum_{i \in Y} \frac{AIM(i)}{CC((i-1),i)}}{|Y_{u,v}|}$, where CC is defined for a pair of classes as the number of distinct edges over the number of all edges between them. As we move away from a node, the dependence be-

comes smaller by calculating the differences of AIM across a selected path in the graph. In addition, we penalize dependence dividing by the distance of the two examined nodes. The corresponding algorithm calculates the dependence of the adjacent nodes expanding progressively the range until it reaches a specified number of nodes. Then it links the nodes to be added using three approximations: a) *CHINS*, which in essence starts with a partial solution consisting of a single schema graph node and then finds one additional node each time; b) *Shortest Paths*, which proceeds similar to CHINS but starts with all nodes available in the summary in the first partial solution and c) *Dependent Paths*, which uses the nodes already visited when computing *Dependence*.

Zoom: Differently, we focus on zooming, 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 size of the given summary.

The simplest approach for zooming-in/out, is to calculate from scratch the $TOP_{n'}$ schema nodes 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: a) *Zoom-in*: Remove the nodes in $TOP_n \backslash TOP_{n'}$ and their connections without recalculating the Steiner-Tree algorithm for $TOP_{n'}$ - this might leave additional nodes in the result summary. b) *Zoom-out - CHINS*: Add the nodes in $TOP_{n'} \backslash TOP_n$ and link them with existing summary using the CHINS approximation algorithm. c) *Zoom-out - Shortest Paths*: Add the nodes in $TOP_{n'} \backslash TOP_n$ and link them with existing summary using the *Shortest Paths* approximation algorithm.

3 Architecture & Demonstration

The high-level architecture of the system is shown in Figure 1 (left). The summarization process starts by uploading an RDF/S file, or by providing the URL of an online file. The file is then stored to a Virtuoso triple store. Our engine preprocesses the available information and stores statistical and metadata information in a different Virtuoso graph. As long as existing information is available for a specific file, it can be reused and not recomputed.

To demonstrate the functionalities of the RDFDigest+, shown in Figure 1 (right)), we will use 5 KBs: the BIOSPHERE ontology, the Financial ontology, the Aktors Portal ontology, the CIDOC-CRM ontology and DBPedia. The variety on the size, the domain and the structure of these ontologies offers an interesting test case for our demonstration. The demonstration will start by selecting one of the aforementioned ontologies and producing 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 the 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 ontology understanding. Further explo-

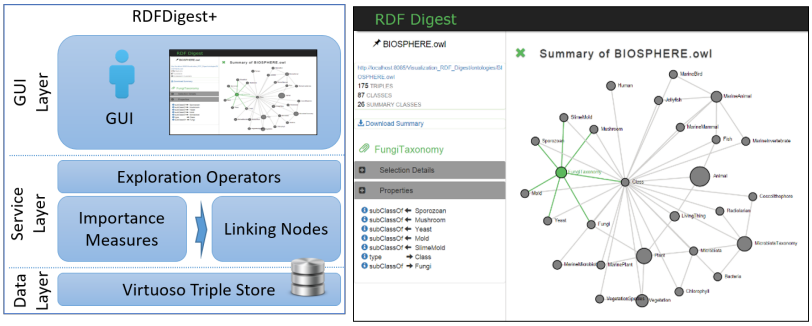

Fig. 1. System architecture (left), Extend and zoom operators (right).

ration 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. Double-clicking on a node extends the summary of that specific node providing more detailed information regarding the dependent nodes. In addition, the summary can be zoomed-in and zoomed-out to present more detailed or more generic information regarding the whole summary. Finally, the user can download the summary as a valid RDF/S document.

The demonstration will proceed in four phases: a) *Selecting important nodes*. The whole idea of exploring RDF/S KBs through summaries will be described and the basics behind each importance measure will be highlighted. According to the user decisions, the results will be discussed. b) *Selecting linking algorithms*. We will demonstrate the differences between the two linking algorithms, focusing on different perspectives for linking the most important nodes. c) *Exploration operators*. We will demonstrate how the two operators help to focus in more detailed or coarser information. d)*Hands-on phase*. Conference participants can directly interact with the system, exploring through summaries the aforementioned ontologies. To our knowledge, no other system today is available on the Web, enabling the summarization of RDF/S KB, providing functionalities for active data exploration through summaries.

References

1. 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.
2. A. Pappas, G. Troullinou, G. Roussakis, H. Kondylakis, and D. Plexousakis. Exploring importance measures for summarizing RDF/S KBs. In *ESWC*, 2017.
3. 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.
4. G. Troullinou, H. Kondylakis, E. Daskalaki, and D. Plexousakis. Ontology understanding without tears: The summarization approach. *Semantic Web*, 8(6):797–815, 2017.
5. G. Troullinou, H. Kondylakis, K. Stefanidis, and D. Plexousakis. Exploring RDF/S KBs using summaries. In *ISWC*, 2018.