Coverage-Based Summaries for RDF KBs

By

Giannis G. Vassiliou

A thesis submitted in partial fulfillment of
the requirements for the degree of

Informatics Engineering

Hellenic Mediterranean University

Heraklion, 2021

Approved by

Papadakis Nikolaos, Associate Professor HMU

Marakakis Manolis, Professor HMU

Dr. Kondylakis Haridimos, Collaborating Researcher, FORTH-ICS

To Giorgos and Rafailia, my loving children

ACKNOWLEDGMENTS

I would like to thank my advisor Associate Professor Nikos Papadakis for his help in completing and writing this thesis.

I also would like to thank Professor Manolis Marakakis for his suggestions and comments.

Further I would like to thank Georgia Troullinou for the excellent cooperation we had.

Finally, I would also like to thank Dr. Haridimos Kondylakis for his guidance and the valuable ideas he shared with me through the whole procedure for starting, completing and writing this thesis.

HELLENIC MEDITERRANEAN UNIVERSITY

ABSTRACT

Coverage-Based Summaries for RDF KBs

As more and more data become available as linked data, the need for efficient and effective methods for their exploration becomes apparent. Semantic summaries try to extract meaning from data, while reducing its size. State of the art structural semantic summaries, focus primarily on the graph structure of the data, trying to maximize the summary's utility for query answering, i.e. the query coverage. In this thesis, we present four algorithms, trying to maximize the aforementioned query coverage using ideas borrowed from result diversification. The key idea among all algorithms is, instead of focusing only to the "central" nodes, to push node selection also to the perimeter of the graph. Our experiments show the potential of our

algorithms and demonstrate the considerable advantages gained for answering larger fragments of user queries.

Table of Contents

List of Figures

1 Introduction

The rapid explosion of the available data in the web has led to an enormous amount of widely available RDF datasets. However, these datasets often have extremely complex and large 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 [1], [8].

One method for condensing and simplifying such datasets is through semantic summaries. According to our recent survey [1], a semantic summary is a compact information, extracted from the original RDF graph, intuitively; summarization is a way to extract meaning from data while reducing its size, and/or a graph, which some applications can exploit instead of the original graph to perform certain tasks more efficiently.

Structural summaries focus first and foremost on the graph structure, respectively the paths and sub-graphs one encounters in the RDF graph. State of the art works in the area of structural summarization [8], [9] first try to identify the most important nodes of the schema graph, and then to optimally link those, producing a connected schema sub-graph. As such,

the size of the presented schema graph is reduced to a minimum size, so that end users are easier to understand the contents of the generated summary, while in parallel the most important nodes are selected and presented to the user.

The problem. The problem with the state of the are structural semantic summaries is that the selected, most important nodes, are in most of the cases nodes located centrally to the schema graph, missing exploration opportunities for the nodes are located at the perimeter of the graph. To this direction, result diversification has also attracted considerable attention as a means of enhancing the quality of the exploration results presented to the users, as it offers, intuitively more informative results than a homogeneous result [3]. However, to the best of our knowledge, those ideas, although notably useful and interesting, have not yet migrated into structural semantic summaries.

Contribution. In this thesis, we focus on summaries that try to maximize query coverage, exploiting ideas from the result diversification field. More specifically, we provide four diverse algorithms for selecting the nodes in the summary to be presented to the user:

- *Node selection based on topology*: Based solely on the structure/topology of the schema graph, we select the nodes with the

maximum shortest path distance. We name the corresponding algorithm LSP as it focuses on the Longest Shortest Paths.

- *Node selection based on importance*: The idea here is to maximize both the topological diversity and the importance of the selected nodes. As already mentioned, structural semantic summarization methods, first provide an ordering of the nodes in terms of importance and then select the top-k nodes to be used for the summary. Our idea here is to select one by one the individual nodes and each time to exclude all its neighbors in distance *r* from the ordering, trying to get representatives from other "importance neighborhoods" as well.
- *Hybrid node selection*: The idea is to combine semantic and structural diversity in order to further improve the generated summary, by starting from the nodes with the longest shortest paths and eliminating the nodes in the ranking within a specific radius. Two variations have been created, the 1- LSP-DisC, and the LSP-DisC, starting with different sets of nodes from the ones with the longest shortest paths as a seed, and progressing to the list of the most important nodes for selecting the remaining ones.

- We experimentally evaluated our approach using real world datasets (DBPedia and SWDF) and show that the produced summaries surpass current state of the art in terms of quality.

To the best of our knowledge, we are the first to incorporate notions from result diversification in structural semantic summaries. Although, in the first two cases, we adapt existing algorithms in our setting, we progress even further producing two novel, hybrid node selection algorithms outperforming other approaches.

The rest of this thesis is structured as follows: In **Section 2**, we present preliminaries required for understanding the remaining of this thesis, whereas related work is presented in **Section 3**. **Section** 4 defines coverage-based summaries and **Section 5** presents the individual algorithms. Finally, **Section 6** presents evaluation and **Section 7** concludes this thesis and presents directions for future work.

2 Preliminaties

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

2.1 RDF

An RDF graph (in short a graph – figure 1) is a set of triples of the form (*s, p, o*). A triple states that a subject s has the property p, and the value of that property is the object o. We consider only well-formed triples, as per the RDF specification belonging to $(U \cup B) \times U \times (U \cup B \cup L)$ where *U* is a set of Uniform Resource Identifiers (URIs), L a set of typed or untyped literals (constants), and *B* a set of blank nodes (unknown URIs or literals); *U*, *B*,*L* are pairwise disjoint. Blank nodes are essential features of RDF allowing to support unknown URI/literal token. As described above, it is easy to see that any RDF graph is a labeled graph. However, as we explain below, RDF graphs may contain an ontology, that is, a set of graph edges to which standard ontology languages attach a special interpretation. The presence of ontologies raises specific challenges when summarizing RDF graphs, which do not occur when only plain data graphs are considered.

Notations. We use *s*, *p*, and *o* as placeholders for subjects, properties and objects, respectively. The RDF standard has a set of built-in classes and properties, as part of the rdf: and rdfs: pre-defined namespaces. We use these namespaces exactly for these classes and properties, e.g., rdf:type specifies the class(es) to which a resource belongs.

2.2 RDF Schema (RDFS)

RDFS allows enhancing the assertions made in an RDF graph with the use of an ontology, i.e., by declaring semantic constraints between the classes and the properties they use.

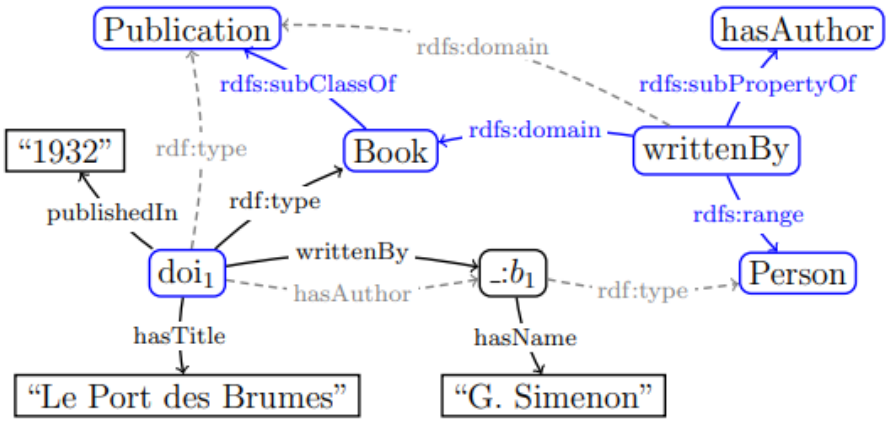

Figure 1 - RDF graph and its implicit triple

An example is shown in Figure 1, where we see the classes Book, Publication and DOI (for the description of a book), where Book is subclass

of Publication and “has” DOI. A specific book instance for DOI is “Le port des Brumes” (title) and “1932” (published), “G.Simenon”(writtenBy). A Book is also “writtenBy” a “Person”, and whatever is “writtenBy” also “has Author”.

2.3 SPARQL

SPARQL on the other hand, is the standard W3C query language used to query RDF graphs. We consider its popular conjunctive fragment consisting of Basic Graph Pattern (BGP) queries. BGP queries are also the most widely used in real-world applications. A BGP is a generalization of an RDF graph in which variables may also appear as subject, property and object of triples.

```
PREFIX type: <http://dbpedia.org/class/yago/>
PREFIX prop: <http://dbpedia.org/property/>
SELECT ?country_name ?population
WHERE {
    ?country a type:LandlockedCountries ;
             rdfs:label ?country_name ;
             prop:populationEstimate ?population .
    FILTER (?population > 15000000 && lang-
Matches(lang(?country_name), "EN")) .
} ORDER BY DESC(?population)
```

An example SPARQL query using DBPedia, which finds all landlocked countries with a population greater than 15 million, with the highest population country first, is shown below.

3 Related work

Summarization has been applied to RDF data to extract concise and meaningful information from RDF knowledge bases, representing their content as faithfully as possible. There is no single concept of RDF summary, and not a single but many approaches to build such summaries; each is better suited for some uses, and each presents specific challenges with respect to its construction. RDF summarization has been used in multiple application scenarios, such as id visualization to get a quick understanding of the data. It should be noted that indexing, query optimization and query evaluation were studied as standalone problems in the data management areas, before the focus went to semantic RDF graphs; therefore, several summarization methods initially studied for data graphs were later adapted to RDF. Among the currently known RDF summarization approaches, some only consider the graph data without the ontology, some others consider only the ontology. Finally some use a mix of the two. Summarization methods rely on a large variety of concepts and tools, comprising structural graph characteristics, statistics, pattern mining or a mix thereof. Summarization methods also differ in their usage scope.

Some summarize an RDF graph into a smaller one, allowing some RDF processing (e.g., query answering) to be applied on the summary (also). The output of other summarization methods is a set of rules, or a set of frequent patterns, an ontology etc.

Based on a recent survey [1] An RDF summary is one or both among the following:

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

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

3.1 Works on summaries

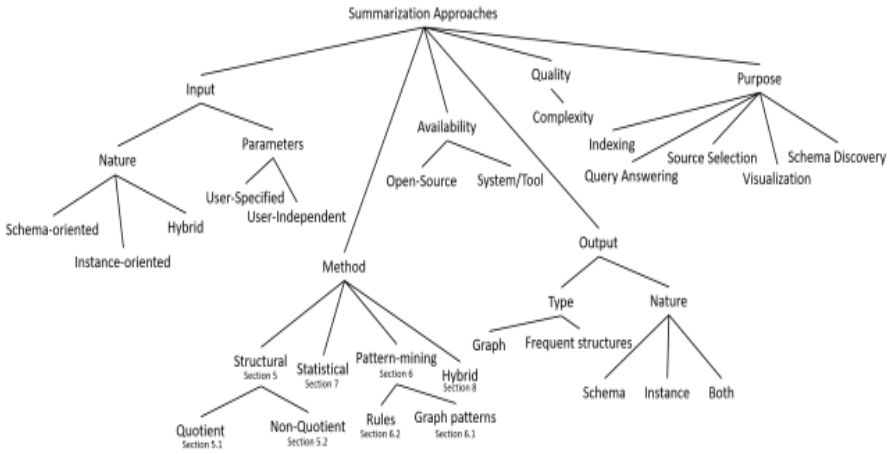

Figure 2 – A taxonomy of the works in the area (as presented in [1])

Based on the methods employed (fig.2), we have four main trends in summary creation.

Structural methods exploit the graph structure, respectively the paths and the sub-graphs one encounters in the RDF graph. They can further classified to the *quotient* methods that consider some notion of "equivalence" identifying node's representatives and *non-quotient* that mostly select individual nodes out of the RDF graph.

Patter mining methods, employ mining techniques for discovering patterns in the data, the summary is then built out of the patterns identified.

Statistical methods: These methods summarize the contents of a graph quantitatively. The focus is on counting occurrences, such as counting class instances or building value histograms per class, property and value type; other quantitative measures are frequency of usage of certain properties, vocabularies, average length of string literals etc.

Hybrid methods: To this category belong works that combine structural, statistical and pattern mining techniques.

As our approach is a non-**quotient structural** summarization approach we will focus next only on the works from that specific area.(figure 3)

3.1.1 Returning only the nodes

Peroni et al. [6] and Wu et al. [11] focused on non-quotient structural summarization. The former tries to automatically identify the key con-

cepts in an ontology combining cognitive principles, lexical and topological measurements such as the density and the coverage. The algorithm created is evaluated against results produced by human experts. In the latter the authors use similar algorithms to identify the most important concepts and relations in an iterative manner. Most precisely the incorporate a custom algorithm named CARRank (figure 4) trying to rank important concepts and relations in the ontology simultaneously evaluating against user specified important nodes (concepts) and relationships in a case study and comparing with other ranking algorithms.

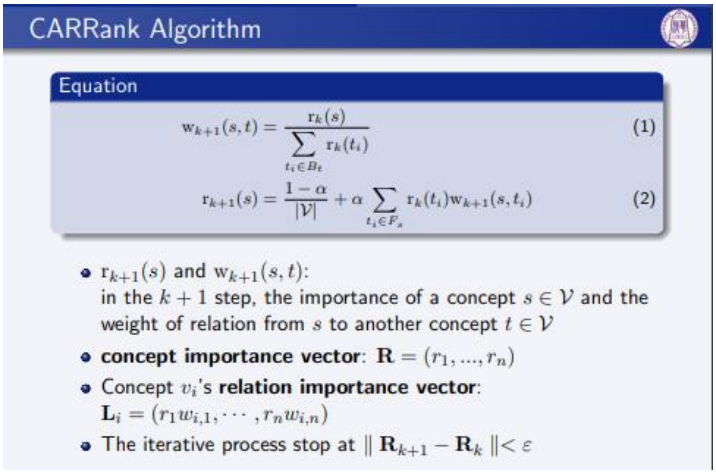

Figure 3- The CARRank Algorithm

However, both of these efforts focus only on returning the most important nodes and not on returning an entire graph summary.

3.1.2 Trying to extract sentences

Zhang et al. [12] propose the use of RDF sentence as the basic unit of summarization. Using a user provided ontology, it is mapped to a set of RDF sentences (figure 4) by extracting those using specific algorithmic strategies, and using them as units for the summary creation.

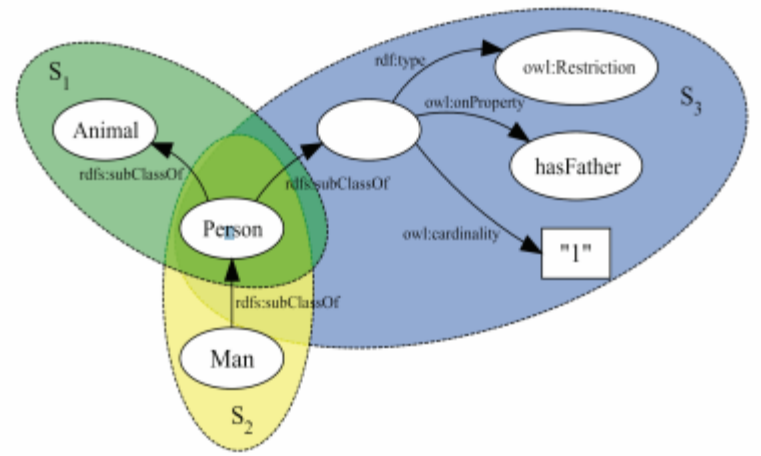

Figure 4 - Creating Sentences

Using measures such as the degree-centrality, the betweenness and the eigenvector centrality, the salience of a RDF sentence is assessed. Collecting the RDF Sentences leads to the creating to the RDF Sentence Graph, which provides the links between the RDF sentences.

3.1.3 Combining with user preferences

In Queiroz-Sousa et al. [7] the authors try to combine user preferences with the degree centrality and the closeness to calculate the importance of a node and then they use an algorithm to find paths that include the most important nodes in the final graph. “Two tasks are needed to build an ontology summary: identify the key concepts and select them in order to produce a subontology of the original ontology. The first task is accomplished using relevance measures and user-defined parameters whilst the second one is performed by the Broaden Relevant Paths (BRP) algorithm, proposed to identify the best path (ontology summary) in a graph (ontology)that represents a set of interrelated vertexes (concepts). However, the corresponding algorithm prioritizes direct neighbors ignoring that the selection of other paths that could maximize the importance of the selected summary. (figure 5)

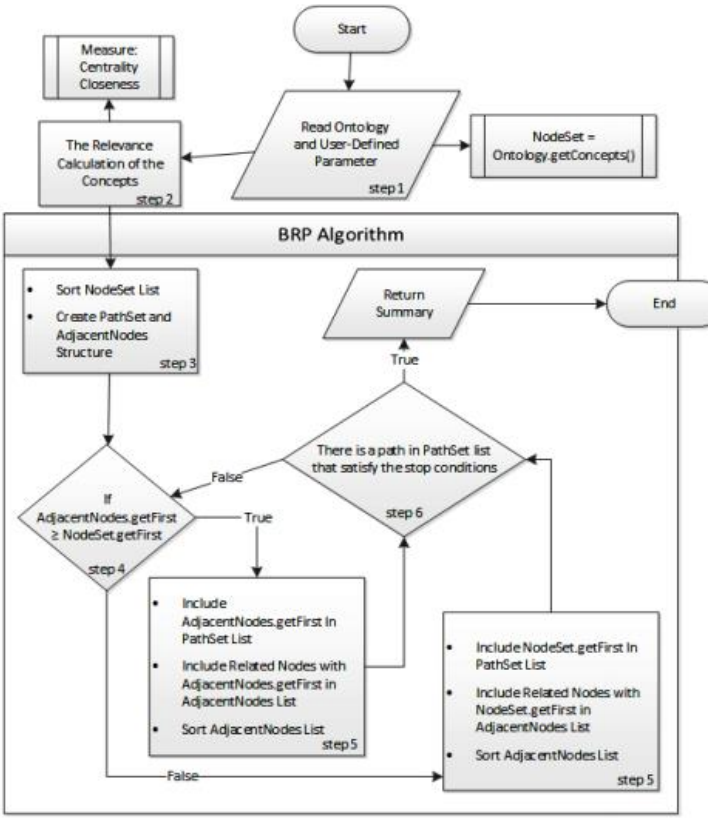

Figure 5 - BRP-Ontology summarization

3.1.4 RDFDigest+

Finally, RDFDigest+ (figure 6) proposes the betweenness centrality for effectively constructing summaries, and show that the generated summaries dominate other existing approaches in the area [8], [5].

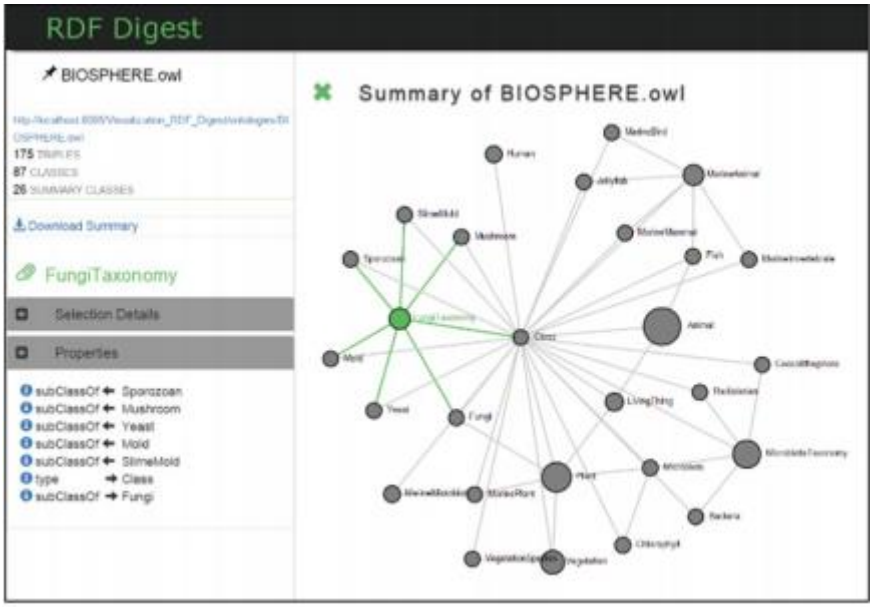

Figure 6 - The RDFDigest System

3.1.5 A comparison of the structural summaries

In figure 7 we present some of the main structural summaries approaches. We can see that some of them are using as input both the schema and the instances graph and others just the schema. Some have as target only visualization of the final (reduced) schema and others cope with query answering. Some have as output the final schema (a connected graph) and others present only the selected nodes finally.

Work	RDF input component	Input requirements	Purpose	Output type	Output Nature	System Theory
RDFDigest	Instance and Schema	Required schema, Parameterized user input, RDF/OWL, Semantics-aware, Handle implicit data	Visualization, query answering tasks	Labeled graph	Schema	System
Queiroz et al.	Schema	Required schema, Parameterized user input, RDF/OWL	Visualization	Labeled graph	Schema	System
RDF Sentence Graph (Zhang et al)	Schema	Required schema, Parameterized user input, RDF/OWL	Visualization	Labeled Graph	Schema	System
Peroni at al.	Schema	Required schema, Parameterized user input, RDF/OWL	Visualization	Nodes	Nodes	
Wu et al.	Schema	Required schema, Parameterized user input, RDF/OWL	Visualization	Nodes	Nodes	

Figure 7 – Works on non-quotient, structural RDF summaries

3.1.6 Relation to graph summaries

Since our focus is on RDF graph summarization techniques, we leave out of our scope graph summarization techniques tailored for other classes of graphs, e.g., biological data graphs , social networks etc. We focus on techniques that have either been specifically devised for RDF, or adapted to the task of summarizing RDF graphs. Nevertheless our approach can be directly applied to generic graph summaries as well, as the centralities measures and the notions of diversity adopted, hold for generic summaries as well.

3.2 Comparison with our approach

The state-of-the-art work regarding Structural Summaries is RDFDigest+. In a way, it produces the summaries with the best results of the finally created schema graph regarding the coverage of the benchmark queries nodes.

However, as we will show both analytically and experimentally in the sequel, RDFDigest+ focuses on selecting only centrally located nodes, missing the opportunity to exploit also nodes in the perimeter of the graph, for improving the quality of the generated summaries. So by moving the summary nodes' selection (by using result diversification) away from the central nodes, we have a way to exploit an area that was not

explored before and give the user another "view" of the initial graph which (as we will prove) competes and even surpasses in benchmark queries nodes' coverage the state-of-the-art works.

3.3 Works on result diversification

For our summary creating algorithms we used ideas from result diversifcation which is a topic which explores results retrieved by user queries to improve the results quality. The widely used diversification models are MAXMIN and MAXSUM, which aim at selecting a subset S of P so that the minimum or the average pairwise distance of the selected objects is maximized [13, 14, 15]. Another way to achieve diversification of results is the DisC algorithm proposed by Drosou et al. [3] "A DisC diverse subset of a query result contains objects such that each object in the result is represented by a similar object in the diverse subset and the objects in the diverse subset are dissimilar to each other." Describing the algorithm from the original thesis: "Let P be the set of objects in a query result. We consider two objects p1 and p2 in P to be similar, if $dist(p1, p2) \leq r$ for some distance function dist and real number r, where r is a tuning parameter that we call radius. Given P, we select a representative subset $S \subseteq P$ to be

presented to the user such that: (i) all objects in P are similar with at least one object in S and (ii) no two objects in S are similar with each other. The first condition ensures that all objects in P are represented, or covered, by at least one object in the selected subset. The second condition ensures that the selected objects of P are dissimilar. We call the set S r-Dissimilar and Covering subset or r-DisC diverse subset."

So far and to the best of our knowledge there are no current works using diversification ideas for RDF schema summarization. In our case the implementation of the DisC algorithm shifts the selection of the nodes away from the center and pushes the node selection out to the perimeter of the RDF schema, trying to diversify the result without neglecting the fact that the central nodes could be useful for our summary.

4 Coverage-based summaries

Here, we will follow an approach similar to [8] and [9], 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 $V = <G_S, G_I, \lambda, \tau_c>$ 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 $V_S \subseteq C \cup L$.
- 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 $V_I \subseteq I \cup L$.
- A labelling function $\lambda: E_S \cup E_I \rightarrow 2^P$ determines the property URI that each edge corresponds to (properties with multiple domains/ranges may appear in more than one edge).
- A function $\tau_c : I \rightarrow 2^C$ 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 $v_1, v_2 \in V_S$, such that, $\lambda(e) = p$. In addition, for brevity, we will call schema node a node $s \in V_S$, class node a node $c \in C \cap V_S$, and instance node a node $i \in I \cap V_I$. 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.

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 select the schema nodes for generating the summary, and (ii) how to link selected nodes in order to produce a valid subschema graph.

However, independent of the way we select the schema nodes, we can safely assume the existence of a function *select_nodes*(G_S), that for a given schema graph G_S, returns the k nodes to participate in the summary. Then, we can define a summary schema graph of size k to be the following:

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

- $V'_S = select\_nodes(G_S) \cup V_{ADD}$, V_{ADD} represents the nodes in the summary used only to link the nodes in *select_nodes*(G_S)
- $\forall v_i, v_j \in select\_nodes(G_S), \exists path(v_i, v_j) \in G'_S$,
- $\nexists$ summary schema graph $G''_S=(V''_S, E''_S)$ including the nodes in *select_nodes*(G_S), such that, $|V''_S|<|V'_S|$.

According to the aforementioned algorithm having a set of nodes selected by the *select_nodes* function, we are looking then the minimum number of the additional nodes to be introduced out of the initial schema graph in order to link those selected nodes. This is due to the fact that, introducing many additional nodes shifts the focus of the summary and

decreases summary's quality. As such, we model the problem of linking the most important nodes as a variation of the well-known Graph Steiner-Tree problem (GSTP) [10]:

Definition 3. (The Graph Steiner-Tree problem (GSTP)) Given an undirected graph $G_S = (V_S, E_S)$, with edge weights w: $E_S \rightarrow R+$ and a node set of terminals $select\_nodes(G_S) \subseteq V_S$, find a minimum-weight tree $T \in G_S$ such that $select\_nodes(G_S) \subseteq V_T$ and $E_T \subseteq E$.

In our case, we ignore as well the direction in the edges, we assume equal weights for all edges, whereas the node set of terminals are the schema nodes selected by the *select_nodes* function. The corresponding algorithm, targets at minimizing the additional nodes introduced for connecting the selected nodes. However, the problem is NP-hard, and as such the CHINS approximation algorithm is used in our case [5].

Next, we focus on how to implement the function $select\_nodes(G_S)$. Ideally, we would like to select important, informative nodes that are best suited to describe the contents of the entire KB, maximizing the summary's utility *for query answering*. State of the art structural semantic

summaries so far, adopt centrality measures to identify the most important nodes.

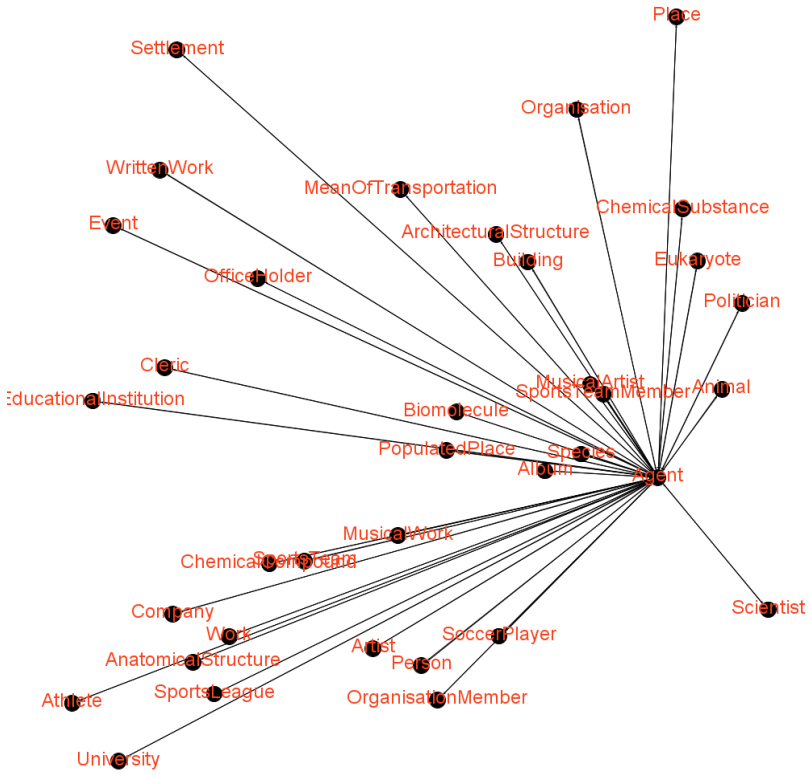

Figure 8. - A summary from DBpedia as produced by RDFDigest+

An example is shown in figure 8 from RDFDigest+, that exploits betweenness centrality. We can see that the summary focuses on a central part of the entire graph. Although such a summary would be really useful for queries around the *Agent* class (in the center), for a non-homogeneous

query workload a summary like the one presented on the figure 9 would arguably be better.

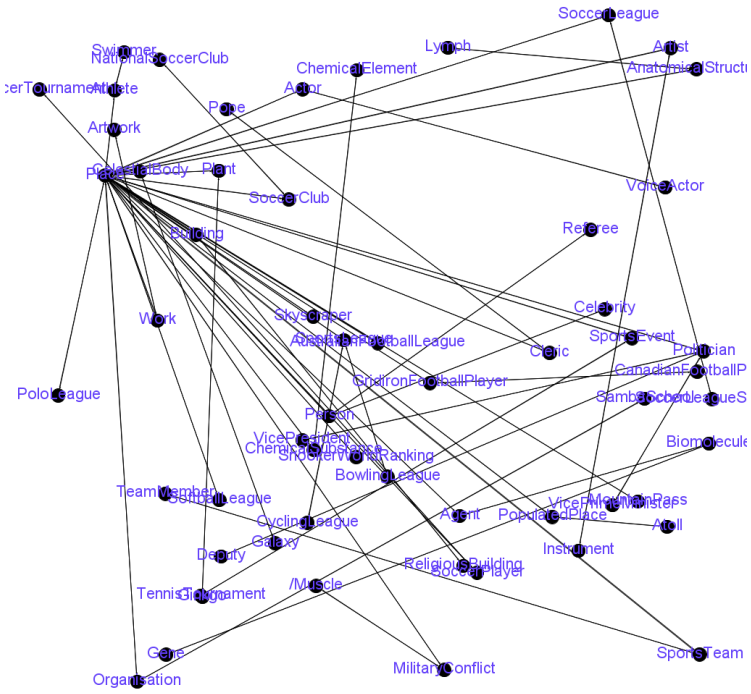

Figure 9. - A summary from DBpedia as produced by 1-LSP-DISC

As such, assuming a query log, we would like to maximize **the fragments of queries that are answered by the summary**. More specifically, having a summary, we can calculate for each query that can be partially be answered by the summary, the percentage of the classes and properties that are included in the summary, i.e. the success nodes and

the success properties. 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.

Definition 4. (Coverage). Assuming a graph G_S = (Vs, E_S), a query workload Q=$\{q_1, \ldots, q_n\}$, and two weights for nodes and edges, i.e. w_{nodes} and w_{prop}, we define coverage as follows:

$$Coverage(Gs, Q) = AVG_{q=i}^{n}\left(w_{nodes}\frac{success\_nodes(q_i)}{nodes(q_i)} + w_{prop}\frac{success\_properties(q_i)}{properties(q_i)}\right)$$

As our summaries are node-based (they are generated based on the selected 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). Now, having defined the coverage for a given query workload we can describe a coverage-based summary schema graph:

Definition 5. (Coverage-based Summary Schema Graph of size *k*). Assuming a query workload *Q*, a coverage-based summary schema graph of size *k*, is a *summary schema graph of size k* maximizing the coverage for the queries in *Q*.

Ideally, we would like to product summaries without exploiting a query log for their construction, as in most of the cases those query logs are not available (we were only able to collect query logs with thousands of real user queries only for a few hand-picked datasets). As such the target of our thesis is to generate heuristic algorithms that can lead to the same result without exploiting user queries for constructing the summaries.

4.1 Methodology for creating a summary

To create the schema summaries (figure 10), we get as input the original schema graph. Every algorithm of ours is producing the nodes to be included in the final schema graph (of course the number of the nodes to be included is given as a parameter by the user). We use the Graph Steiner Tree algorithm to connect the possible unconnected nodes we have from the previous step and thus have the final schema summary as a connected graph.

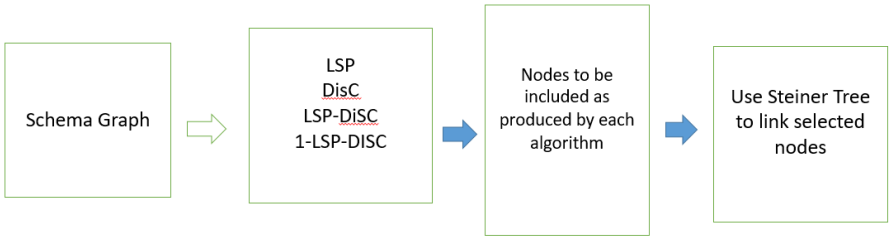

Figure 10 - Summary Creation Workflow

5 Constructing Coverage-Based Summaries

In this thesis, we implement four, diverse algorithms for constructing coverage-based summaries. The first one (LSP) is solely based on the topology of the graph completely ignoring other aspects, the second one (DiSC) focuses is an adaptation of an algorithm for result diversification [3], exploiting individual nodes' importance, whereas the LSP-DisC, 1-LSP-DisC are our own contributions combining topology and importance in order to select schema nodes. In the sequel, we present the corresponding algorithms in detail.

5.1 The LSP Algorithm

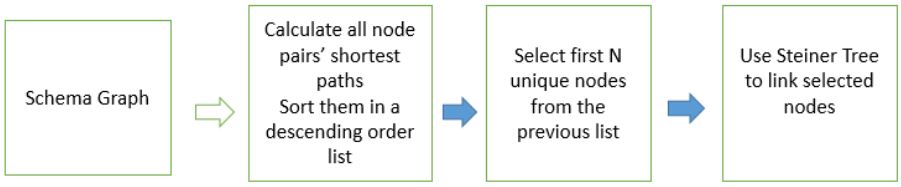

Figure 11 - LSP based summary creation – Targets the perimeter nodes of the graph

The Longest Shortest Paths (LSP) algorithm, shown in algorithm 1, calculates all shortest paths between the schema nodes in the schema graph (line 1), ranks them based on the calculated distance in a descending order (line 2) and then selects the first k nodes from that list (line 3). The complexity of Algorithm 1 is $O(m|Vs|)$ and mainly comes from calculating all pairs shortest paths, where $|Vs|$ is the number of nodes in G_S and m a quantity based on $|Vs|$ as shown in [2].

Algorithm 1: LSP(G_S,k)

Input: A schema graph G_S, k the number of schema nodes to select.

Output: A set of schema nodes N.

1. LSP := Calculate_all_pairs_shortest_paths(G_S)
2. Sort (LSP) in descending order based on the distance
3. N := Select top-k nodes from LSP
4. **Return** N

5.2 The DisC algorithm.

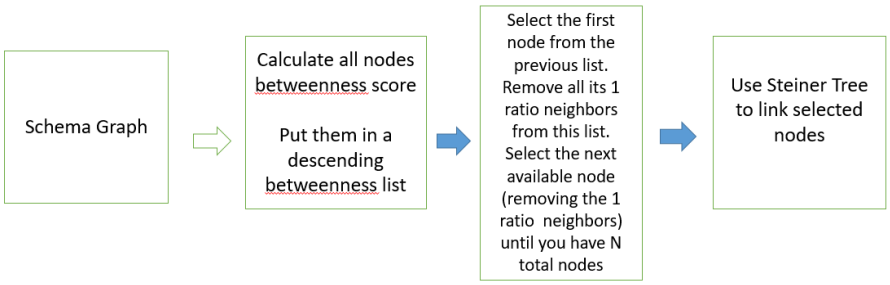

Figure 12 – DisC based summary creation – Diversifies the central nodes' selection

The DisC algorithm is based on a related approach [3], introduced for result diversification. The main idea here is not only to select nodes based on an importance measure but in each selection step to exclude its neighbors (in a radius *r*). As such, node selection is guided to select nodes out of the entire graph instead of selecting only the central ones (which usually importance measures find as more important).

The corresponding algorithm is shown in Algorithm 2. Initially we calculate for each node its centrality measure (lines 2-3). As previous works on structural semantic summaries have identified that the betweenness centrality has an excellent performance for producing summaries that optimize query answering, we are reusing it here as well.

Then we sort them based on their centrality measure in descending order (line 4). Then we get the first node of the list (the most important one based on its centrality measure (line 6) and, and we exclude from the list all its neighbors in a distance *r* (line 7). We repeat the node selection and the corresponding exclusions of the neighbors, till out list is empty or we have selected the *k* nodes required (line 5). Finally, we return the selected nodes to the users.

Algorithm 2: DisC(G_S,*k*, *r*)

Input: A schema graph G_S, *k* the number of schema nodes to select, *r* the radius of the nodes to be excluded.

Output: A set of schema nodes *N*.

1. *N*:= ∅
2. **for each** *node* **in** G_S **do**
3. *betweennes*[*node*]:=calculate_ betweennees(G_S)
4. sort_nodes(*betweennes*)
5. **while** *betweennes* != ∅ and |*N*| < *k* **do**
6. **Add** *top node* in *betweennes* to *N*
7. **Remove** *node* neighbors in a radious *r* from the *betweennes* list
4. **Return** *N*

For calculating the betweennes for all nodes we need O(|*Vs*|+|*Es*|), then for sorting the nodes in a descending order based on their centrality O(|*Vs*|) and then O(|*Vs*|) for constructing the final list by removing each node's neighbors, assuming appropriate adjacency structures. As such, the overall complexity of our algorithm is O(|*Vs*|+|*Es*|).

5.3 The LSP-DisC Algorithm.

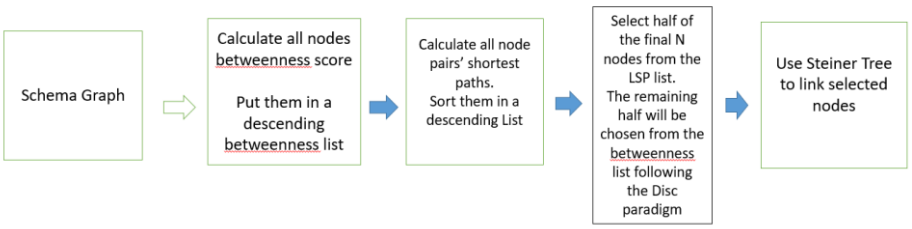

Figure 13 - LSP-DisC summary creation – a way to combine perimeter nodes and diversified central nodes

To select schema nodes using the LSP-DisC algorithm, we combine the LSP and the DisC algorithms. The corresponding algorithm is shown in Algorithm 3. At the beginning, again, we calculate the betweenneess of all nodes (lines 3-4) and we sort the nodes in a descending order based on their betweenness (line 5). Then we calculate all pairs shortest paths (line 6) and we sort them in descending order based on their distance (line 7). Then we select the *m* unique nodes from the LSP list (in our test

case; 50% of the expected nodes). We add them in the final list and eliminate them from the descending betweenness list along with their *r*-hop neighbors (lines 9-11). We continue by choosing the first available node from the betweenness list that has not been deleted so far, adding it in the final list until we have the total number of required nodes. Obviously, the complexity of the algorithm is $O(m|Vs|)+ O(|Vs|+|Es|) \leq O(|Vs|+|Es|)$

Algorithm 3: LSP-DisC(G_S,*k*, *r*)
Input: A schema graph G_S, *k* the number of schema nodes to select, *r* the radius of the nodes to be excluded.

Output: A set of schema nodes *N*.

1. *N*:= ∅
2. N_{LSP}:= ∅
3. **for each** *node* **in** G_S **do**
4. *betweenness*[*node*]:=calculate_ betweennees(G_S)
5. sort_nodes(*betweenness*)
6. *LSP* := Calculate_all_pairs_shortest_paths(G_S)
7. Sort (*LSP*) in descending order based on the distance
8. N_{LSP} := Select top-*m* nodes from *LSP*
9. **for each** node in N_{LSP} **do**
10. **Add** *node to N*
11. **Remove** *node and node's* neighbors in a radious *r* from the *betweenness* list

12. **while** *betweenness* != ∅ and |*N*| < *k* **do**

13. **Add** *top node* in *betweenness* to *N*

14. **Remove** *node* neighbors in a radious *r* from the *betweenness* list

15. **Return** *N*

5.4 The 1-LSP-DisC Algorithm

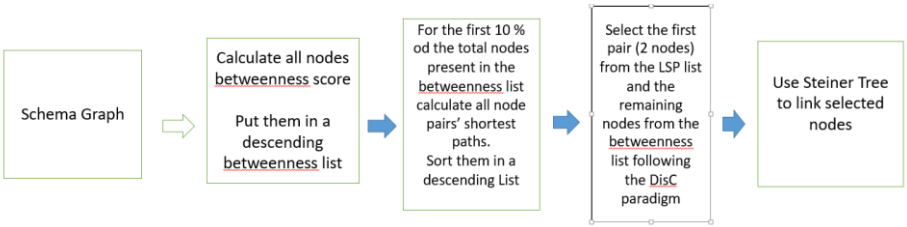

Figure 14 - 1-LSP-Disc summary creation – focusing on a central part of the graphand utilizing again the LSP and DisC algorithms

Another algorithm trying to optimally select schema nodes is the 1-LSP-DisC algorithm, presented in Algorithm 4. The algorithm constructs the ordered list of schema nodes based on their betweenness (lines 3-5). Then we select the top-m nodes from that list (in our case about 10-15% of the total nodes number) and we calculate all pair shortest paths for those m schema nodes (line 6). Then we select the two most distant nodes

(lines 7-8), adding them to the list of the selected nodes and removing their r-hop neighbors (lines 9-11). Then the algorithm proceeds as the previous two algorithms by selecting the next more important node in the ordered list and removing each time its r-hop neighbors (lines 12-14). Then the result nodes are returned to the users. Again, the complexity of the corresponding algorithm is $O(m|V_S|)+ O(|V_S|+|E_S|) \leq O(|V_S|+|E_S|)$

Algorithm 4: 1-LSP-DisC(G_S,k, r)
Input: A schema graph G_S, k the number of schema nodes to select, r the radius of the nodes to be excluded.

Output: A set of schema nodes N.

1. N:= ∅
2. N_{LSP}:= ∅
3. **for each** *node* **in** G_S **do**
4. *betweenness*[*node*]:=calculate_betweenness(G_S)
5. sort_nodes(*betweenness*)
6. $N_{BETt:}$=Select top-m nodes from betweenness
6. LSP := Calculate_all_pairs_shortest_paths(N_{BET})
7. Sort (LSP) in descending order based on the distance
8. N_{LSP} := Select top-2 nodes from LSP
9. **for each** *node* in N_{LSP} **do**
10. **Add** *node to N*

11. **Remove** *node and node's* neighbors in a radious *r* from the *betweenness* list

12. **while** *betweenness* != ∅ and |*N*| < *k* **do**

13. **Add** *top node* in *betweenness* to *N*

14. **Remove** *node* neighbors in a radious *r* from the *betweenness* list

15. **Return** *N*

5.5 Conclusion

We implemented four algorithms trying to focus on no central nodes of the graph. We used ideas from result diversification and combine them with structural ideas like choosing distant nodes with LSP. Some of the produced summaries focus on central (but diverse) nodes, some on perimeter nodes and some try to combine the view from both worlds.

6 Evaluation

In this section, we present the evaluation performed for the implemented algorithms.

Setup. All experiments reported in this section were performed in a laptop computer running Windows 10 with an Intel i5 3320M- 2.6 GHz CPU and 8GB of main memory. All algorithms and were implemented in Java JDK 8.

Competitors. We contrast our results with the state-of-the-art approach on structural summaries, the RDFDigest+ [9] and report our findings. Our goal is to assess the added value of our implemented algorithms on maximizing query coverage. In addition, we have to node that in our case the radius was set to one for DisC-based algorithms, as this was the only case were we could get in the summary the 10% of the available nodes (for r>1, many neighbors were excluded from the list and as such only a few nodes were eventually left).

Datasets. For evaluating our approach, we use DBpedia and Semantic Web Dog Food (SWDF) KBs. **SWDF** consists of 120 classes, 72 properties and more than 300K triples. We use a query log containing 902 user queries provided by SWDF SPARQL end-point for this dataset version. **DBpedia v3.8** consists of 422 classes, 1323 properties and more

than 2.3M instances, and offers an interesting use-case for exploration. To identify the quality of our approach, we use a query log containing 56K user queries provided by the DBpedia SPARQL end-point for the corresponding DBpedia version.

6.1 Results on the SWDF dataset

For evaluating our algorithms on the SWDF dataset, we request a summary with 16 summary schema nodes (~10% summary). As the Steiner Tree algorithm for linking the selected schema nodes introduces additional nodes in the summary, at the end, the final summary includes 16 nodes in the case of RDFDigest+, 22 nodes in the case of DisC, 22 nodes in the case of LSP, 19 nodes in the case of LSP-DisC and 20 nodes in the case of 1-LSP-DisC.

Already this is an indication that our coverage-based summaries select more distant nodes, which requires more nodes to be introduced for connecting those into a summary schema graph.

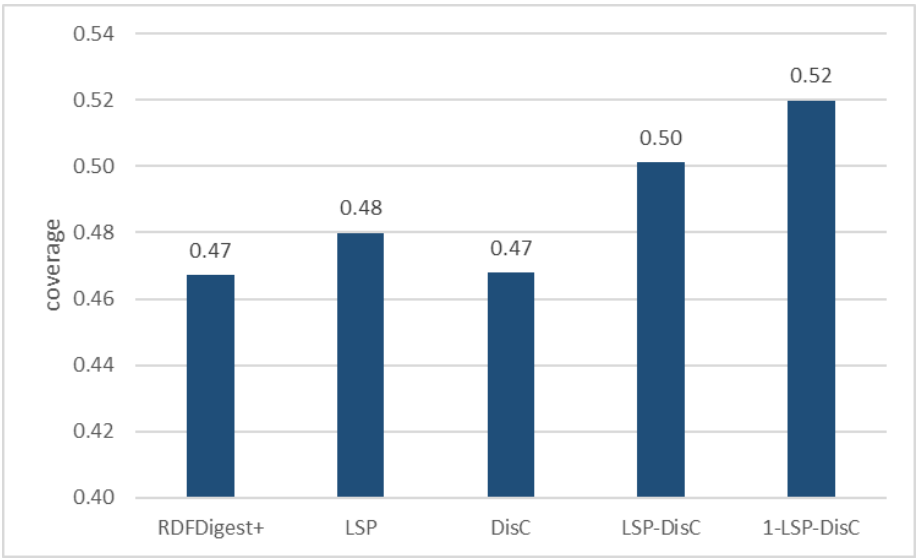

Figure. 15. - Coverage for the various algorithms for the SWDF dataset.

Based on the summaries generated by the various algorithms, we calculate next the coverage for the available queries. We have to note that in this dataset all schema graph nodes are queried at least once in every user query. The results are shown in 6. As shown, the RDFDigest+ generates a summary with a coverage of 47 %, whereas all coverage-based algorithms, besides DisC outperform RDFDigest+. In essence, most of our algorithms are able to answer larger fragments of user queries, resulting in a better summary quality. The 1-LSP-DisC method has the best coverage (52%), showing the benefits of adopting it, for subsequent query answering.

6.2 Evaluation on DBpedia

Next we move to a different, more challenging dataset, DBpedia. Here we request a summary based on 36 nodes using the corresponding algorithms (~10% summary). Again, the Steiner Tree algorithm for linking the selected schema nodes introduces additional nodes in the summary. At the end, the final summary included 36 nodes in the case of RDFDigest+, 61 nodes in the case of DisC, 47 nodes in the case of LSP, 51 nodes in the case of LSP-DisC and 58 nodes in the case of 1-LSP-DisC.

Similarly to SWDF, already this is an indication, that our coverage-based summaries select more distant nodes, which requires more nodes to be introduced for connecting those into the summary schema graph.

Moving our attention to the DBpedia queries, we observe that large parts of the schema graph are completely ignored by the users. More specifically from the 422 schema nodes of the DBPedia, only 177 of them appear in the queries whereas the remaining nodes do not. Figure 7 presents the results for the various algorithms for the coverage. As shown, again 1-LSP-DisC outperforms RDFDigest+. Looking at the produced summaries, the ones produced by RDFDigest+ tend to include more popular nodes whereas our coverage-based summaries are able to include

nodes also on the perimeter, offering a more diverse view of the schema graph. However, as most of the user queries are ignoring a large fragment of the available nodes, this has an impact on the summaries produces by our coverage-based algorithms as well.

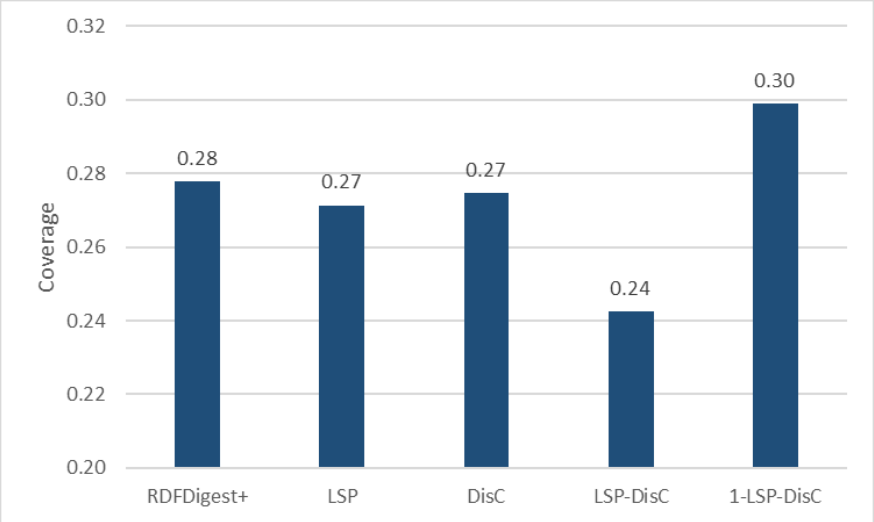

Figure. 16. - Coverage for the summaries generated by the various algorithms for DBpedia.

6.3 Efficiency

Next, we evaluate efficiency for the various algorithms presented in the thesis. More specifically, for the two datasets in our evaluation, we produce the 10% summaries using the various methods and we report the

average of ten executions. The results are shown in Fig 8 for the SWDF dataset and in Fig.9 for the DBpedia dataset.

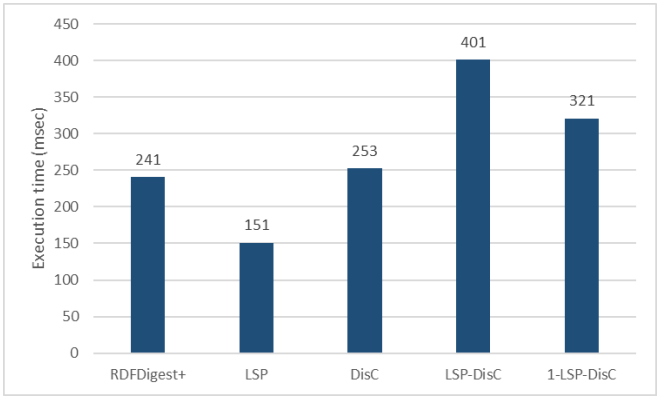

Figure. 17. - Execution time for the SWDF dataset.

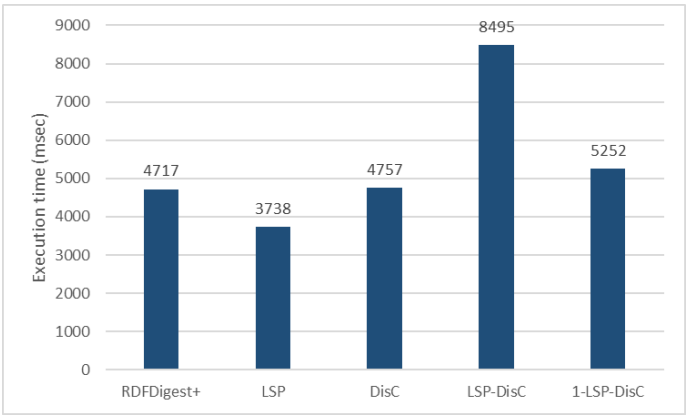

Figure 18 Execution time for DBPedia

As shown selecting the nodes for the SWDF is one order of magnitude faster in all cases than DBpedia. This is reasonable as DBpedia's schema graph is about four times larger than SWDF.

In addition, we can see that LSP is the fastest algorithms of all, independent of the fact that it has to compute all pairs shortest paths. This is reasonable, however, and in accordance to the complexity of the various algorithms, as the remaining algorithms have to compute the betweenness, which is a really costly procedure. RDFDigest+ has only to compute the betweenness whereas the other algorithms have to do additional computations. In addition, as shown, the most time-consuming algorithms are the hybrid ones as they involve both the creation of the betweenness list and the calculation of all pairs shortest paths. The same applies for the 1-LSP-DisC, however here the all pairs shortest paths are only calculated for a subset of the schema nodes (the top-m ones).

7 Conclusions & Future Work

In this thesis, we focus on producing summaries that maximize their utility for query answering. We explore ideas from result diversification and we present four diverse algorithms for constructing structural semantic summaries, pushing nodes' selection to the perimeter of the graph trying to collect representative nodes from both each topological and importance "neighborhood" of the graph. Our experiments confirm that the produced summaries indeed maximize the coverage of thousands of user queries, although they have been constructed, without using the specific workload.

7.1 Future Work

As next step, we intend to explore personalized summaries that will be constructed based on an initial node selection performed by the user. The seed nodes could come from text resources or specific queries based on which the users would like to explore the graph. The fact is that building a non-personalized summary (like the ones in this work), you can never be sure whether the particular summary could be useful to a specific user case, because you do not know the interests of the user. On the other side, the current work can be valuable when building personalized summaries

when combining the interest of the user that could be in the perimeter of the graph, so a “diverse” view of the summary could be very promising.

7.2 Potential Extensions to Generic Graphs

Our implementation can be directly extended to generic graphs besides RDF/S graphs as our algorithms exploit centrality measures and ideas from the diversity domain that both have been initially proposed for generic graphs.

References

1. Cebiric, S., Kondylakis, H., Kotzinos, D., Manolescu, I., Troullinou, G., Zneika, M.: Summarizing semantic graphs: a survey. VLDB J., 2019:28(3): 295-327.
2. Chan, T.M. All-pairs shortest paths for unweighted undirected graphs in o (mn) time. ACM Transactions on Algorithms (TALG) 8.4, 2012, 1-17.
3. Drosou, M., Pitoura, E.: DisC diversity: result diversification based on dissimilarity and coverage. Proc. VLDB Endow, 2012, 6(1): 13-24.
4. Kondylakis, H., Kotzinos, D., Manolescu, I.: RDF graph summarization: principles, techniques and applications. EDBT, 2019, 433-436.
5. Pappas, A., Troullinou, G., Roussakis, G., Kondylakis, H., Plexousakis, D.: Exploring Importance Measures for Summarizing RDF/S KBs. In ESWC, 2017, 387-403.
6. Peroni, S., Motta, E., d'Aquin, M.: Identifying key concepts in an ontology, through the integration of cognitive principles with statistical and topological measures. In Asian Semantic Web Conference (ASWC), 2008, 242-256.
7. Queiroz-Sousa, P.O., Salgado, A.C., Pires, C.E.: A method for building personalized ontology summaries. Journal of Information and Data Management, 2013, 4(3):236.
8. Troullinou, G., Kondylakis, H., Stefanidis, K., Plexousakis, D.: Exploring RDFS KBs Using Summaries. International Semantic Web Conference (1) 2018: 268-284.
9. Troullinou, G., Kondylakis, H., Daskalaki, E., Plexousakis, D.: Ontology understanding without tears: The summarization approach. Semantic Web 2017, 8(6): 797-815.
10. Voß, S.: Steiner's problem in graphs: Heuristic methods. Discrete Applied Mathematics, 1992, 40(1):45–72.
11. Wu, G., Li, J., Feng, L., Wang, K.: Identifying potentially important concepts and relations in an ontology. In International Semantic Web Conference (ISWC), 2008, 33-49.
12. Zhang, X., Cheng, G., Qu, Y.: Ontology summarization based on RDF sentence graph. WWW, 2007, 707–716.
13.] S. Gollapudi and A. Sharma. An axiomatic approach for result diversification. In WWW, 2009.
14. A. Borodin, H. C. Lee, and Y. Ye. Max-sum diversifcation, monotone submodular functions and dynamic updates. In PODS, 2012.
15. M. R. Vieira, H. L. Razente, M. C. N. Barioni, M. Hadjieleftheriou, D. Srivastava, C. Traina, and V. J. Tsotras. On query result diversification. In ICDE, 2011