iSummary : Workload-based, Personalized summaries for RDF/S KBs.

Fanis Alevizakis

Thesis submitted in partial fulfillment of the requirements for the

Masters' of Science degree in Computer Science and Engineering

University of Crete
School of Sciences and Engineering
Computer Science Department
Voutes University Campus, 700 13 Heraklion, Crete, Greece

Thesis Advisors:
Professor *Dimitris Plexousakis*
Dr. *Haridimos Kondylakis*

This work has been performed at the University of Crete, School of Sciences and Engineering, Computer Science Department.

The work has been supported by the Foundation for Research and Technology - Hellas (FORTH), Institute of Computer Science (ICS).

University of Crete
Computer Science Department

iSummary : Workload-based, Personalized summaries for RDF/S KBs.

Thesis submitted by
Fanis Alevizakis
in partial fulfillment of the requirements for the
Masters' of Science degree in Computer Science

THESIS APPROVAL

Author: _____
Fanis Alevizakis

Committee approvals: DIMITRIOS PLEXOUSAKIS Digitally signed by DIMITRIOS PLEXOUSAKIS Date: 2022.03.03 12:58:09 +02'00'
Dimitris Plexousakis
Professor, Thesis Supervisor

KONSTANTINOS MAGOUTIS Digitally signed by KONSTANTINOS MAGOUTIS Date: 2022.03.03 12:50:29 +02'00'
Kostas Magoutis
Professor, Committee Member

GEORGIOS FLOURIS Digitally signed by GEORGIOS FLOURIS Date: 2022.03.02 12:50:12 +02'00'
Giorgos Flouris
Principal Researcher, Committee Member

Departmental approval: _____
Polyvios Pratikakis
Associate Professor, Director of Graduate Studies

Heraklion, February 2022

iSummary : Workload-based, Personalized summaries for RDF/S KBs.

Abstract

The explosion on the size and the complexity of the available RDF data has led to the need for efficient and effective methods for their understanding and exploration. To this direction, semantic summaries have been proposed for extracting compact information from the original RDF graph that many applications can exploit instead of the original RDF graph, offering a way to quickly explore and understand the contents of the various sources. In this thesis we present iSummary, a novel approach for constructing personalized summaries based on query workloads. The main idea behind our approach is to exploit knowledge captured in existing user queries for identifying the most interesting resources and to capitalize query workloads for constructing and presenting high quality personalized summaries. We demonstrate our approach using two real world datasets and the corresponding query workloads and we show the advantages of our approach.

iSummary : Workload-based , Προσωποποιημένες συνόψεις για RDF/S KBs δεδομένα

Περίληψη

Η αύξηση του μεγέθους και της πολυπλοκότητας των διαθέσιμων RDF δεδομένων έχει οδηγήσει στην ανάγκη εύρεσης έξυπνων και αποτελεσματικών μεθόδων για την κατανόηση και την εξερεύνηση τους. Σε αυτή τη κατεύθυνση οι συνόψεις γράφων έχουν σκοπό να εξάγουν συνοπτική πληροφορία από τους αρχικούς γράφους που πολλές εφαρμογές μπορούν να εκμεταλλευτούν αντί των αρχικών, για την περαιτέρω διερεύνηση των διαθέσιμων πηγών. Σε αυτή την εργασία παρουσιάζουμε το iSummary, μία προσέγγιση με την οποία δημιουργούνται προσωποποιημένες συνόψεις βασισμένες σε επερωτήσεις των χρηστών. Η κύρια ιδέα πίσω από την προσέγγιση μας είναι να χρησιμοποιήσουμε την πληροφορία που ήδη υπάρχει σε υπάρχοντα ερωτήματα χρηστών έτσι ώστε να εντοπίσουμε τα πιο σημαντικά κομμάτια του γράφου, δημιουργώντας και παρουσιάζοντας υψηλής ποιότητας προσωποποιημένες συνόψεις. Αξιολογούμε την προσέγγισή μας χρησιμοποιώντας δύο πραγματικές βάσεις γνώσης και τα αντίστοιχα σετ επερωτήσεων και δείχνουμε τα πλεονεκτήματα της προσέγγισής μας.

Acknowledgements

Αρχικά, θα ήθελα να ευχαριστήσω θερμά τον επόπτη καθηγητή μου κ.Χαρίδημο Κονδυλάκη για την άψογη συνεργασία και ουσιαστική συμβολή του στην ολοκλήρωση της παρούσας μεταπτυχιακής εργασίας,καθώς και την Γεωργία Τρουλινού Διδακτορικό φοιτητή υπό την επίβλεψη του κ.Κονδυλάκη.Ακόμη θέλω να εκφράσω τις ευχαριστίες μου στον κ. Δημήτρη Πλεξουσάκη,στον κ. Κώστα Μαγκούτη καί τον κ. Γιωργο Φλουρή για την προθυμία τους να συμμετέχουν στην τριμελή επιτροπή. Τέλος θα ήθελα να ευχαριστήσω την οικογένεια μου και τους φίλους μου για την συμπαράσταση και την υποστήριξη που μου έδωσαν όλα αυτά τα χρόνια.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Daily, a tremendous amount of new information becomes available online. RDF Knowledge bases rapidly grow to include millions or even billions of triples which are offered through the web. For example, the Linked Open Data Cloud, currently includes more than 62 billion triples, organized in large and complex RDF data graphs.

The complexity and the size of those data sources limit their exploitation potential and necessitate effective and efficient ways to explore and understand their content. To this direction, semantic summarization has been proposed as a way to extract useful, minimized information out of large semantic graphs that many applications can exploit instead of the original data graphs for performing certain tasks more efficiently such as visualization, exploration, query answering etc [4]. Structural semantic summaries focus mostly on the structure of the graph for extracting the required information, whereas non-quotient structural semantic summaries try to select the most important parts of the graph for generating the result summaries.

The problem. Most of the existing works in the area of structural, non-quotient semantic summaries, generate generic static summaries [4]. However as different persons have different data exploration needs the generated summaries should be tailored specifically to individual's interests. Although this has already been recognized by the research community, the approaches offering personalized summaries so far, rely on parts (or weights) selected initially by the users, then followed by algorithms making various vague assumptions about the relevant subsets out of the semantic graph that should complement user initial choices.

The solution. In this work, starting from nodes selected by the user we exploit query workloads in order to identify the most relevant parts of the semantic graph, as previous users have already identified through their queries, in order to formulate the generated summaries. More specifically:

- We enable users to select the set of nodes of interest that they would like the generated summaries to be based on.
- Based on these nodes we find the nodes that co-occur with the selected nodes in user queries and rank them by their frequency. The main assumption here is that the most frequent the co-occurence of a node with the selected nodes, the most relevant will be according to the existing query workload. Based on their ranking then we select top-k nodes to participate in the constructed schema summary.
- In order to link the selected top-k frequent nodes with user selections and to construct a connected subgraph out of the initial graph we again exploit the queries by finding the most frequent connections between user selections and the nodes in the top-k. As such, each query is transformed to a mini-graph, traversed in order to find the most frequent, minimum path linking user choices with each one of the top-k nodes.
- As in many cases we only have a limited set of queries we cannot find concrete paths without variables - remember than in each triple of the queries usually we have at least a variable. In that case, we execute the queries over relevant SPARQL endpoints, in order to substitute variables with their mappings from the graph.
- We experimentally evaluate our approach in two real datasets and the corresponding workloads, i.e. WIKIDATA and DBPedia, showing the benefits of our approach maximizing coverage for user queries.

To the best of our knowledge this is the first approach constructing personalized, structural, non-quotient semantic summaries exploiting query workloads.

The rest of this thesis is organized as follows: Chapter 2 provides the preliminaries for better understanding this work and Chapter 3 presents related work. Then Chapter 4 elaborates on our methodology for generating summaries. Chapter 5 presents the experimental evaluation of our work. Finally Chapter 6 concludes this thesis and presents directions for future work.

Chapter 2

Preliminaries

In this chapter we present background theory for RDF/S and RDF graphs.

2.1 Graphs: Core Concepts

Labeled directed graphs RDF datasets can be seen as labelled directed graphs. Further, most of the proposals for summarizing an RDF graph also model the summary as a directed graph. As such, given a set A of labels we denote by G=(V,E) an edge labeled directed graph with E the edges and V nodes. An example of a labeled direct graph is shown in Figure 2.1 whose vertices are V, and whose edges are $E \subset V$ x A x V where A edge labels are attached to the edges.

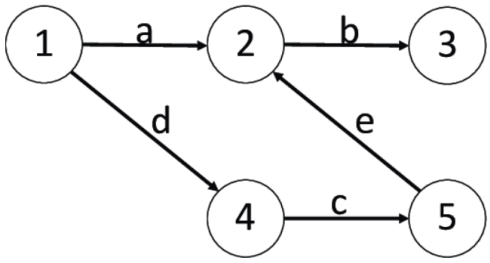

Figure 2.1: Labeled direct graph

2.2 RDF & RDF Schema

RDF graph. In this work, we focus on datasets expressed in RDF, as RDF is among the widely-used standards for publishing and representing data on the

Web. Those datasets are based on triples of the form of (s p o) which record that *subject* s is related to *object* o via predicate p. Formally, representation of RDF data is based on three disjoint and infinite sets of resources, namely: URIs (U), literals (L) and blank nodes (B). A key concept for RDF is that of URIs or Unique Resource Identifiers; these can be used in either of the s, p and o positions to uniquely refer to some entity, relationship, or concept. Literals (constants) are also allowed in the o position. Blank nodes in RDF, allow representing a form of incomplete information through standing for unknown constants or URIs. As such, a triple is a tuple (s p o) from $(U \cup B) \times U \times (U \cup L \cup B)$.

RDF Schema. RDF Schema (Resource Description Framework Schema, RDFS, RDF(S), RDF-S, or RDF/S) is the accompanying W3C proposal of a schema language for RDF. It is used to describe classes and relationships between classes (such as inheritance). Further, it allows specifying properties, and relationships that may hold between pairs of properties, or between a class and a *property*. RDFS statements are also represented by triples. The statements allowed by RDFS are presented in Figure 2.3.

RDFS statements	meaning
C rdf:type rdfs:Class	entity C in class RDF
P rdf:type rdf:Property	entity P is property RDF
R rdf:type C	the entity R is instance of class C
C rdfs:subClassOf C'	the class C is subclass of class C'
P rdfs:subPropertyOf P'	the property P is subProperty of property P'
P rdfs:domain C	all the subjects where link with property P are instances of class C
P rdfs:range C	all objects where link with property P are instances of class C

Figure 2.2: Properties RDFS

Example 1. Now consider as an example, the RDF/S KB shown in Figure 2.3, that includes information on the university domain. The figure visualizes persons and organizations and also presents some indicative instances. Note that prefixes, are omitted from the figure for sake of clarity.

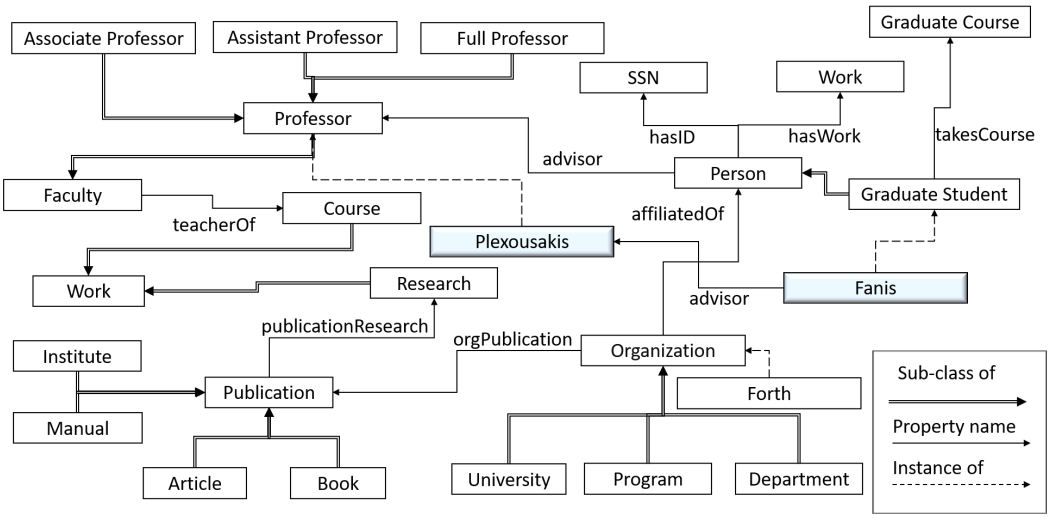

Figure 2.3: Example RDF graph.

Sparql queries. SPARQL, pronounced 'sparkle', is the standard query language and protocol for Linked Open Data on the web or for RDF triple-stores [1]. SPARQL, stands is a protocol and an RDF Query Language and enables users to query information from databases or any data source that can be mapped to RDF. The SPARQL standard is designed and endorsed by the W3C and helps users and developers focus on what they would like to know instead of how a database is organized. SPARQL is in essence a graph-matching language. SPARQL queries contain a set of triples patterns, also called basic graph patterns (BGPs). Triple patterns are like RDF triples that each of the subject, predicate and object may be a variable, a URI or a literal. Solutions to the variables are then found by matching the patterns in the query to triples in the dataset. Thus, SPARQL queries are pattern matching queries on triples, that compose an RDF data graph. A typical syntax of a BGP query is:

$$SELECT\ ?v_1...?v_m\ WHERE\{t_1...t_n\}$$

where $t_1, ..., t_n$ is a set of triple patterns, and $?v_1...?v_m$ are variables used in $t_1, ..., t_n$ that define the output of the query. Join operations are encoded in those queries by sharing the same variable in more than one triple pattern.

According to the position of the variables in the triple patterns, a query can have different shapes. Common types of BGP queries are *star* queries and *path* queries. *Star* are the ones characterized by subject-subject joins between the triple patterns - as the join variable is on the subject position.

[1] `https://www.w3.org/TR/sparql11-query/`

On the other hand, *path* queries are formulated using triple patterns with subject-object (or object-subject) joins. For example, the join variable can be on the object position in one triple pattern, and on the subject position in the other. As *complex*, we characterize queries that combine the aforementioned query-types.

Example 2. Assume also that we have a query workload available that comprises of the three queries presented in Figure 2.3. The first query ask for the persons and their advisors, the second one the persons that are affiliated with one organization whereas the third query asks for all organizations available in the RDF graph.

```
Q1. SELECT ?x ?y WHERE
    {x? a Person. y? a Professor. ?x advisor ?y.}

Q2. SELECT ?x ?y WHERE
    {x? a Person. y? a Organization. ?y affilatedOf ?x.}

Q3. SELECT ?y WHERE
    {y? a Organization.}
```

Figure 2.4: Example workload with three SPARQL queries

Chapter 3

Related Work

3.1 Semantic Summarization, an overview

The explosion on the size of the available RDF data has led to the need to explore query and understand data sources. Due to the complex structure of the RDF graphs, semantic summaries have been proposed to provide effective and efficient solutions to this direction. Semantic summaries try to extract compact information from RDF datasets tha can be used instead of the original graph in order to perform certain tasks more efficiently such as increase the speed of query answering, visualize data, identify appropriate sources for querying etc.

Summarizations techniques can be classified based on their methods to the following large categories:

1. **Structural methods** are those which consider the graph structure, respectively the paths and subgraphs one encounters in the RDF graph. We can further classify the structural methods into the following categories:

   (a) Quotient: Summarizing a graph by assigning a representative to each class of equivalence if the nodes in original graph. A feature of structural quotient methods is that each graph node is represented by exactly one summary node, given that one node can only belong to one equivalence class.

   (b) Non-quotient: Methods in this category try to identify the most important parts of the graph, usually exploiting centrality measures to identify the most important nodes and to interconnect them in the summary. Such methods aim to build an overview of the graph even if some graph nodes may not be represented at all.

2. **Pattern mining methods** In this category all methods works with mining techniques for discovering patterns in the data;the summary is then built out of the patterns identified by mining

3. **Statistical methods** These methods summarizing data of graph quantitatively.This means that count number of class,instances types properties also with these methods make histograms and other graph value types.In addition other measures are frequency of certain nodes,average length of string literals.

4. **Hybrid methods** This method works in combination with structural statistical and pattern-mining techniques.

Figure 3.1 presents dimensions and approaches that need to be explored for the various works in the field based on a relevant figure from [4].

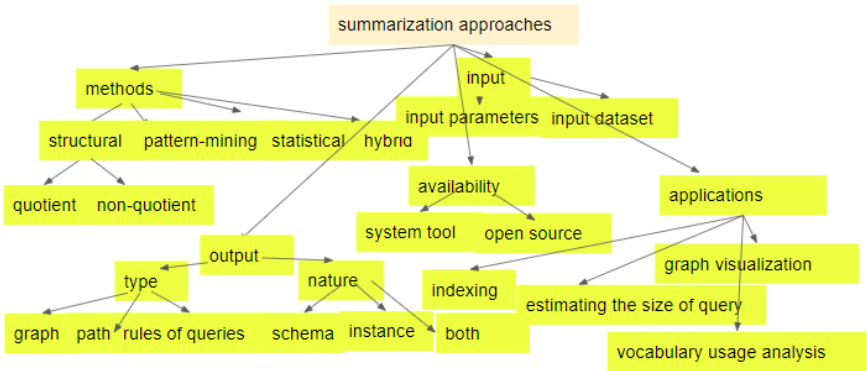

Figure 3.1: Summarization approaches and dimensions

An interesting dimension of the various works in the field is the required **input** of a summary graph. The input can be the input parameters (user specified equivalence relations, maximum summary size, weights assigned to some graph elements etc.) or the input datasets (RDF data graphs are most frequently accepted, usually RDF/S and/or OWL are used for specifying graph). Another interesting dimension is the **output** of the various individual works classifying works in the area based on the output type which differentiates the techniques according to the nature of the final result (summary) that is produced. The summary is sometimes a graph, while in other cases it may be just a selection of frequent structures such as nodes, paths, rules or queries. Another dimension is the **nature**, i.e. whether the output graph is a part of the schema, the instances or both. Another one is the

availability, as it is interesting to know the availability of a given summarization service. Several summarization approaches are made available by their authors as a system/tool or system shared with the public - the implementation of some summarization methods is provided in open source by the authors Further, different summarization works have been developed for different **applications**, such as indexing, partitioning, source selection etc.

3.2 Structural non-quotient summarization techniques

As our approach lies in the structural non-quotient summarization, in the sequel we will provide an overview of the works in the area.

An ontology summarization method focusing on non-quotient personalized summarization is introduced in [13], based on RDF Sentence Graphs. An RDF Sentence Graph is a weighted, directed graph where each vertex represents an RDF sentence, which is a set of RDF Schema statements as shown in Figure 3.2. A link between two sentences exists, if an object of one sentence belongs to another sentence as well. The creation of a sentence graph is customized by domain experts, who provide as input the desired summary size, and their navigation preferences, i.e. weights in the links they are mostly interested in. Then, the importance of each RDF sentence is chosen by determining its centrality in the sentence graph. The authors compare different centrality measures, i.e. the degree, the betweenness, the PageRank, and the HITS scores, and show that weighted in-degree centrality and some eigenvector-based centralities produce better results. Finally, the most important RDF sentences are re-ranked considering the coherence of the summary and the coverage of the original ontology, and the constructed result is returned to the user. This method does not handle implicit information also, it does not consider the instance graph.

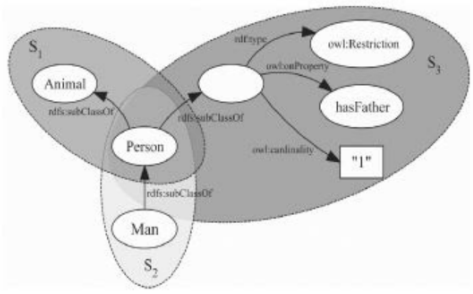

Figure 3.2: Sample RDF sentences

KCE [7, 6] on the other hand, attempts to automatically identify the key concepts in an ontology. To achieve this, it combines cognitive principles with lexical and topological measures (the density and the coverage). The goal is to identify a number of concepts that would be selected by human experts. To this direction a number of criteria are defined based on the idea of the natural category, used for identifying concepts that are information-rich in a psycho-linguistic sense. This is approximated by the following measures:

- Basic level measures how central a concept is in the taxonomy of the ontology. This is combined with the density favouring concepts which have many properties and taxonomic relationships.
- The coverage tires to ensure that no part of the ontology is left unnoticed
- Lastly, the notion of popularity, is based on lexical statistics, and tries to identify concepts commonly used in natural language.

In Queiroz-Sousa et al. [1], on the other hand, 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. The main steps of this summarization method are the following: (i) select the parameters (e.g., the size of the summary and importance thresholds) and possibly nodes that are important according to user's opinion; (ii) compute the relevance of the concepts in the ontology as the weighted sum of the degree centrality and the closeness centrality; (ii) identify the paths linking the selected nodes using the Broaden Relevant Paths algorithm. The specific algorithm tries to find paths of greatest quality within the summarized graph by considering the relevance of the included nodes in the path. The approach supports RDF or OWL ontologies and mainly aims to help ontology understanding through visualization. However, incorporating user preferences is neither explored in detail, nor evaluated.

Wu et al. [12] focused on non-quotient structural summarization. The former tries to automatically identify the key concepts in an ontology combining cognitive principles, lexical and topological measurements such as the density and the coverage. They use an algorithm named CARRank (Concept And Relation Ranking) which is similar with PAGE RANK. CARRank is an automatic ranking algorithm than a specific visualization approach. Four features for potentially important concepts and relations for CARRank are:

1. A concept is more important if there are more relations starting from the concept.

2. A concept is more important if there is a relation starting from the concept to a more important concept.

3. A concept is more important if it has a higher relation weight to any other concept.

4. A relation weight is higher if it starts from a more important concept.

Finally, RDFDigest+ proposes the betweenness centrality for effectively constructing summaries, and experimentally shows that the generated summaries dominate other existing approaches in the area [3],[2]. RDFDigest+ tries to identify user preferences indirectly by combining centrality measures and the numbers of instances for adapting the importance of the schema nodes to be selected. Further RDFDigest+ produces summaries only out of the schema graph, using the instance graph only to calculate weights on the schema nodes.

Our work is the first structural, non-quotient, workload-based personalized summarization method. However, we do not rely on generic centrality measures. Our work accepts minimal user input and then it exploits query workloads for generating high-quality summaries. Further we do not focus only on the schema graph, but we treat both schema and instances the same, considering only their appearance in user queries. We argue that user queries can be more objective than centrality measures on identifying the most relevant parts for a given input by the user.

Chapter 4

Methodology

Semantic summarization aims to highlight the most representative concepts of a dataset, preserving important information and reducing the size and the complexity of the whole data graph. Critical questions for summarization are (i) how to select the nodes for generating the summary, and (ii) how to link selected nodes in order to produce a valid sub-graph out of the original dataset. In this section we initially present the workflow of our approach and we present the corresponding algorithms constructed.

4.1 High-level Workflow

The high-level workflow of our approach is shown in Figure 4.1. As shown iSummary gets as input a) the dataset to summarize (e.g. a SPARQL endpoint containing the RDF graph) b) along with the relevant query workload), c) a initial selection of a resource, and d) the size of the summary in terms of the most important nodes to include. Then it selects the top-k nodes and then it links them based on the query workload and eventually presents a connected graph to the user. For linking the selected nodes, queries are explored by finding the shortest paths out of all queries involving the nodes of interest.

In the sequel we explain in detail the various steps of our workflow and the corresponding algorithms.

4.1.1 Input

In order to select the nodes for our summary, we assume that we have available a query workload $Q = \{q_1, \cdots, q_n\}$ of SPARQL queries. In addition, as we would like to generate a personalized summary, we also receive as input

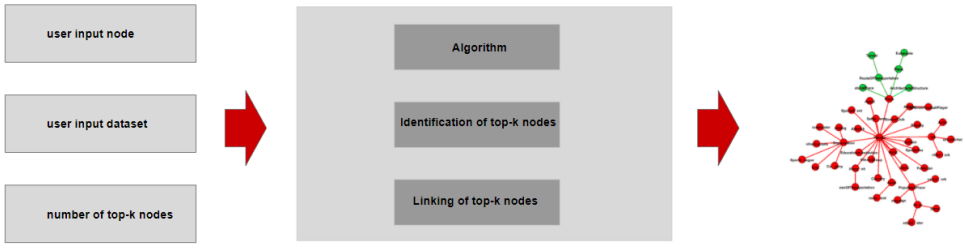

Figure 4.1: Workflow of our summary technique

from the user a node S that s/he would like to focus. Although the user instead of a node can provide a set of nodes as input, in the remaining of this thesis we assume he only provides a single node, without loss of generality for not perplexing the definitions and the presented algorithms. Finally, the number of the most important nodes that will be included in the generated summary should be provided as well (the k in top-k).

4.1.2 Selecting the top-k nodes.

Based on S (input user node) we identify all queries in $Q_S \in Q$ that include that node. Then we calculate the frequency of appearance of the other nodes that exist in Q_S. As such given a workload $Q_S = \{q_{s1}, \cdots, q_{sm}\}$, we construct a node frequency list, where each node is characterized by the number of times appearing in Q_S. Then we order all nodes based on their frequency in Q_S, and we select the top-k ones besides the ones that the user provided as input.

Example 3. Assume that we would like to generate a personalized summary using iSummary for the graph presented in Figure 2.3 exploiting the query workload presented in Figure 2.4. In addition, assume that the user provided as input the node $Person$ to base his personalized summary and we would like to generate a summary adding to the one selected by him, the top-2 most important nodes in the generated summary.

As a first step the queries are filtered keeping only the ones where the node $Person$ appears. As such, $Q3$ is excluded and only $Q1$ and $Q2$ are considered for the remaining phases of the workflow. Based on $Q1$ and $Q2$ the frequency of the nodes that appear in the workload is constructed, shown in Table 4.1. The $Person$ node appears twice in the queries, whereas the remaining nodes, i.e. $Professor$ and $Person$ appear only once. As the user requires a summary with two nodes besides his selection to be included in

Node	Frequency
Person	2
Professor	1
Organization	1

Table 4.1: Frequencies of the example schema nodes.

the summary, all nodes eventually will participate in it.

4.1.3 Linking the top-k nodes.

In order to link the selected nodes we again exploit user queries. The idea here is to link S with the nodes in top-k, one node at a time. For doing that for each $n \in top-k$ we filter the queries in Q_S that include n. Then for each query we construct the mini-graph based on the triple patterns it contains and we retrieve the shortest path in each query for linking a node in S with n. The most frequent shortest path from all queries in Q_S that include n is selected to finally link S with n. Note that in the shortest paths out of the individual queries, variables might appear. If there are paths that do not include variables then we select those. However, if variables appear in all queries, in order to replace the variables with nodes/edges from the original RDF graph, we actually execute the query over the graph to instantiate the variables with real nodes/edges - more specifically, we execute the query and retrieve the first instance returned by the SPARQL endpoint to use for filling the gaps in the path.

Example 4. The algorithm for linking the $Person$ node with the $Professor$ and the $Organization$ nodes will initially try to link $Person$ and $Professor$. As such it will filter the queries in the workload and will only keep the ones where both $Person$ and $Professor$ appear, that is $Q1$. Then the mini-graph based on the triple patterns appearing in Q1 will be constructed in main memory and the Dijktra algorithm will be used to identify the shortest path linking the two nodes. As such the path with the following triple ($Person$, $advisor$, $Professor$) will be included in the summary. Similarly for linking $Person$ with $Organization$ $Q2$ will be used to construct a minigraph in main memory that will provide the path with the following triple ($Organization$, $affiliatedOf$, $Person$). As such eventually the summary consist of the following triples: ($Person$, $advisor$, $Professor$), ($Organization$, $affiliatedOf$, $Person$). The output graph combining the two triples is shown in Figure 4.2.

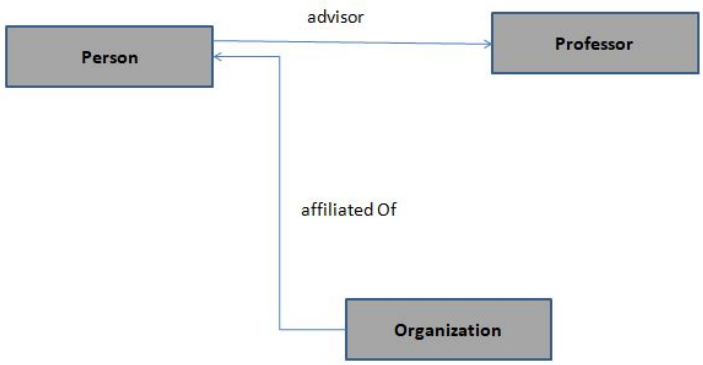

Figure 4.2: Example Summary Graph.

4.1.4 Personalized, Workload-Based summaries

Based on the aforementioned process we are now ready to formally define the personalized, workload-based summaries, assuming a function top_k that given S, G and Q, i.e. $top_k(S, G, Q)$, returns the k most frequent nodes that appear in queries that contain S.

Definition 1 (Personalized, Workload-based Summary of size k)**.** Let $G = (V, E)$ be an RDF graph, Q a query workload and S a node provided by the user. A personalized, workload-based summary of size k, is a graph $G' = (V', E')$, $G' \subseteq G$, such that:

- $V' = \{S\} \cup top_k(S, G, Q) \cup V_{ADD}$,
- $\forall v_i, v_j \in V'$, $\exists path(v_i \rightarrow v_j) \in G'_S$,
- V_{ADD} represents the nodes in the summary used only to link the nodes in $S \cup top_k(S, G, Q)$,
- $\nexists$ Personalized Workload-based schema summary $G''_S = (V''_S, E''_S)$ of size k for V, such that, $|V''_S| < |V'_S|$.

Note that $|V_S|$ depicts the number of nodes included in V_S. In addition there might be multiple paths that can be used to link the selected node with the nodes in top_k. Our algorithm selects lexicographically the first one of the most frequent shortest paths if we assume that they have the same frequency and length.

4.1.5 The algorithm

An algorithm constructing a personalized, workload-based summary of size k is presented in Algorithm 1.

The algorithm starts by receiving as input from the user a node that s/he would like focus on, along with a query workload Q and the number of most important nodes k that he would like to include in the summary.

Then we order all nodes based on their frequency in Q, and we select the top-k ones besides the one that the user provided as input (line 2).

The next step is to link those selected nodes (lines 3-7) with S. More specifically for each node in $x \in top_k$ we filter the queries in Q retrieving $queries_{Sx}$ that contain x and S (line 4). Then for each query in $queries_{Sx}$ we find the minimum path linking S with x (line 5-6). Eventually out of all queries that link x and S we keep the shortest one. In case that we have paths with the same frequency and size we select the first one in the lexicographic order (line 7). Note that if the selected shortest path includes variables then we replace those variables with their mappings from the original graph by executing the corresponding query to the original graph (line 7). As soon as the aforementioned process has been executed for all nodes in top_k we return to the user the constructed graph (line 8).

Algorithm 1 Personalized Workload-based Summary

Input: The input node S, the query workload Q, the number of most important nodes k, the original *dataset*;

Output: The personalized, workload-based summary G

1: $G \leftarrow \emptyset$
2: $top_k \leftarrow selectTop_kNodes(S, Q, k)$
3: **for all** $pairs(S, x), x \in top_k$ **do**
4: $\quad queries_{Sx} \leftarrow findQueriesIncludingNodes(S, x)$
5: $\quad$ **for all** $query \in queries_{Sx}$ **do**
6: $\quad\quad shortestPath[query] \leftarrow getShortestPathFromQuery(query, S, x)$
7: $\quad G \leftarrow G \cup resolveVariables(minimun(shortestPath, dataset))$
8: $return\ G$

The correctness of the algorithm is proved by construction. Although the result produced by the algorithm is deterministic based on its implementation, a personalized workload-based summary might not be unique as many nodes can have the same frequency in the available queries, or there might be available many different shortest paths to connect them.

To identify the complexity of the algorithm we should first identify the complexity of its components. Assuming $|Q|$ the number of queries in the

available workload we first need to scan them once for retrieving the top_k nodes, i.e. $O(|Q|)$. Then for each node in the top_k we need to scan again the queries for filtering them based on the nodes in top_k and S, i.e. $O(k * |Q|)$. Then for each query appearing in the filtering results we should run once the Dijktra algorithm. At the worst case for each k we need to calculate the shortest paths for all queries, k times, i.e. $O(k * |Q| * |V_{Q_S}^2|)$, where $V_{Q_S}^2$ the maximum number of nodes that appear in the queries in the workload. Overall the complexity of the algorithm is

$$O(|Q|) + O(k * |Q|) + O(k * |Q| * |V_{Q_S}^2|) \leq O(k * |Q| * |V_{Q_S}^2|)$$

However, as usually the number of nodes in the queries is limited (usually ≤ 10) we can safely replace $V_{Q_S}^2$ with a constant showing that the algorithm scales linear to the number of queries in the workload. Note also that we assumed that going to the graph to answer a query through the corresponding SPARQL endpoint has a constant time - which might not be true in principle.

Chapter 5

Evaluation

In this chapter, we present the evaluation performed for evaluating our approach using two real world datasets along with the corresponding query workloads. The datasets including the query workloads, along with the source code are available online[1].

5.1 Implementation & Configuration

The iSummary was developed using Java version "1.8.0_281", Java(TM) SE Runtime Environment (build 1.8.0_281-b09) along with Java HotSpot(TM) 64-Bit Server VM (build 25.281-b09, mixed mode). We used the last edition Apache NetBeans IDE 12.5 for developing the corresponding project.

In addition, the evaluation was performed using windows 10 with an Intel® Core™ i5 CPU @ 3.40GHz (8 cores) and 16 GB RAM.

5.2 Datasets

The first dataset we use is the DBpedia version 3.8 dataset along with the corresponding query workload. The query workload consists of 11 separate query log files. The query workload size is 16.3 MB including 58610 queries, whereas the number of total nodes in the query workload (including duplicates) is 120740. The number of edges in the query workload (including duplicates) is 60370. The query workload included queries from real users and as such the queries that were parsed correctly by the Sesame parser were only 6335. The other queries had syntactical errors. The nodes included in the

[1] `https://github.com/fanislevi1994/masterthesis`

correct queries were 14302 and the edges 7151 (including duplicates). Statistics over the DBPedia query workload are presented in Figure 5.1. DBpedia v3.8 consists of 422 classes, 1323 properties and more than 2.3M instances. As we had only a limited number of queries in many cases we had to query the original data graph for getting the mapping of the variables in the queries. For querying we used a local instance of the Virtuoso 6.1.

query workload	dbpedia
query workload size	16.3 MB
number nodes	120740
number edges	60370
number queries	58610
number correct queries	6335
number nodes in correct queries	14302
number edges in correct queries	7151

Figure 5.1: Statistics about the DBpedia query workload.

Similarly to DBPedia, the query workload for WikiData was 10.51 MB retrieved from [5]. The query workload includes 11765 queries with 148356 nodes and 74178 edges (including duplicates). All WikiData queries were successfully parsed using the Sesame query parser as they had been preprocessed. Statistics about the WikiData query workload are provided in Figure 5.2. For querying the WikiData RDF graph whenever needed we resolved to the online WikiData SPARQL endpoint. Wikidata comprises over 4.9 billion triples.

5.3 Metrics

For evaluating the quality of the generated algorithms we use coverage as it has been proved rather useful in evaluating structural, non-quotient semantic summaries in the past [11], [8], [10], [9]. The idea behind coverage is that, ideally, we would like to maximize the fragments of the queries that are answered by the summary. More specifically, a summary that is able to provide answers to bigger and more query fragments from the query workload is preferable. However as we are generating personalized summaries, we

query workload	wikidata
query workload size	10.51 MB
number nodes	148356
number edges	74178
number queries	10454

Figure 5.2: Statistics about the WikiData query workload.

would like the generated personalized summaries to maximize the number and fragments that include the node that the user provided in the input of the algorithm. As such, based on an input node S we define the total coverage as follows:

Definition 2 (Query Coverage)**.** Assuming a query workload Q, an input node S, and two weights for nodes and edges, i.e. w_n and w_p, we define coverage as follows:

$$Coverage(Q, S) = \frac{1}{n}\sum_{i=1}^{n}(w_n\frac{s_{nodes}(q_i)}{nodes(q_i)} + w_p\frac{s_{props}(q_i)}{properties(q_i)}) \tag{5.1}$$

where $\{q_1, \cdots, q_n\}$ the queries in Q that include S.

In our experiments we set w_n=0.8 and $w_p = 0.2$ as in essence our summaries are node-based and we prioritise node selection.

5.4 Coverage for variable k

Next we present our experiments for a variable number of k for both Wikidata and DBPedia. As we are the first to construct personalized workload-based summaries we were not able to find a direct competitor. Nevertheless we made a consistent effort to compare ourselves with other non-workload-based approaches for personalized summaries. However other works were not able to process large graphs such DBPedia and Wikidata for generating a personalized summary. As such we compare our approach with a random baseline, where we select randomly the nodes and the properties from the query workload.

For Wikidata the results are presented in Figure 5.3 and for DBPedia in Figure 5.4. For each experiment for both Random and iSummary we

executed 10 initial random selections of a single node and we present the average coverage.

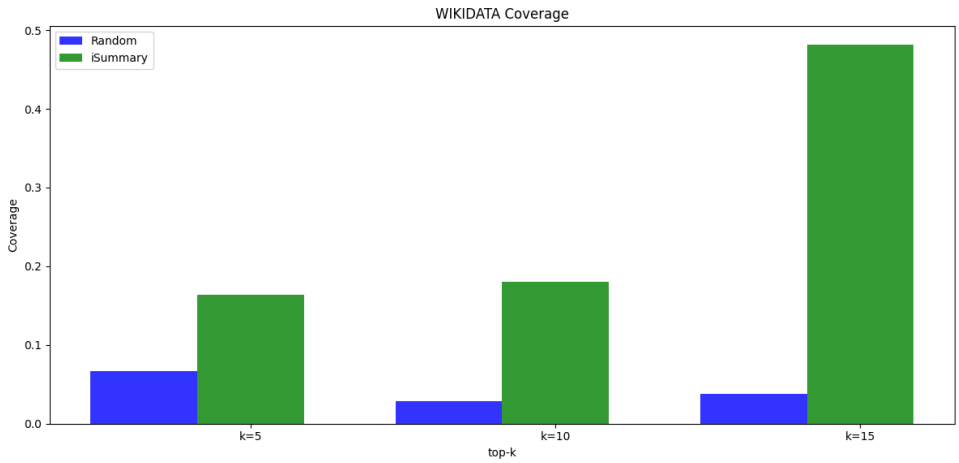

Figure 5.3: Wikidata coverage for various k.

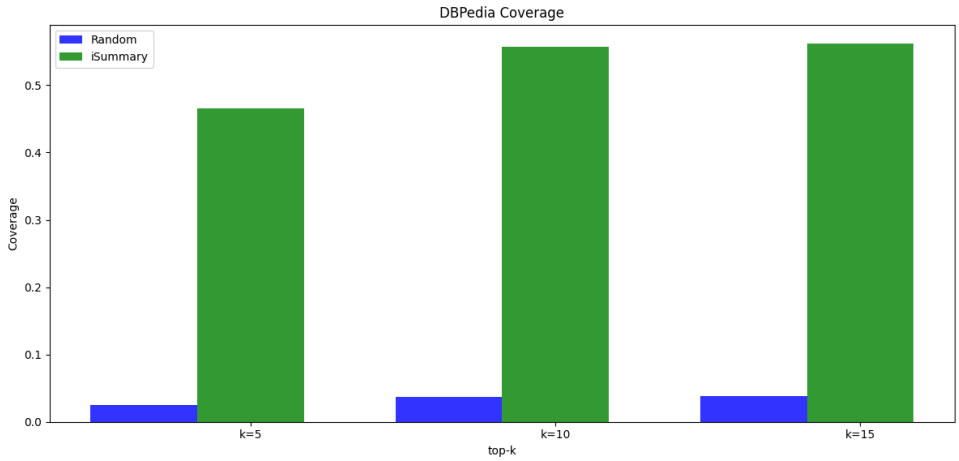

Figure 5.4: DBPedia coverage for various k.

As we can see in all cases our approach has consistently better results than random selection, demonstrating the high-quality of the generated summaries. In fact we can see that as more nodes are selected the coverage of our approach increases as well as more nodes are added to the summary. However, just adding additional nodes is not enough, as only adding high-quality nodes increases the overall coverage. This is evident in our approach, as randomly adding more nodes marginally improves coverage only in some cases.

Note also that as the number of WikiData queries is larger than the number of DBpedia it is reasonable to be a bit more difficult to cover more queries. Nevertheless the algorithm shows a stability among different datasets reaching a coverage of around 0.5 for k=15, meaning that on it covers a significant number of relevant query fragments.

5.5 Splitting a Query Workload into Training & Test

In a next experiment we tried to identify whether our algorithm could be used for constructing the summaries based on a set of queries and then evaluating the coverage on a different set of queries. The experiment was performed for Wikidata as we had enough queries to split them in parts. We randomly divided the queries into two parts, 70% of them for the "training", i.e. for constructing a summary and half of them for "testing", i.e. used only for calculating the coverage.

Again we initially select randomly an input node, and we calculate the summary for k=5, 10, and 15 using the train query workload and then we evaluate the coverage on the test dataset. We execute each experiment two times and we present the mean coverage achieved each time. The results are shown in Figure 5.5. As shown still we are able to achieve a significant coverage for the various cases. In addition, as shown as we increase k the coverage is also increased. As less queries are now available for learning to construct a good summary it is evident that the quality of the generated summary is slightly reduced. This shows that our approach is sensitive to the query workload available as expected.

5.6 Total Number of Nodes in the Summary

In this section we report the number of the nodes in each summary. The results are shown in Figure 5.6. In all cases the number of the final nodes is about two times the k, as additional nodes have to be introduced in order to connect the selected nodes in the top_k.

5.7 Execution Time

Finally in Figure 5.7 we present the execution time for WikiData as we increase k. As shown the more nodes in the summary the more time it

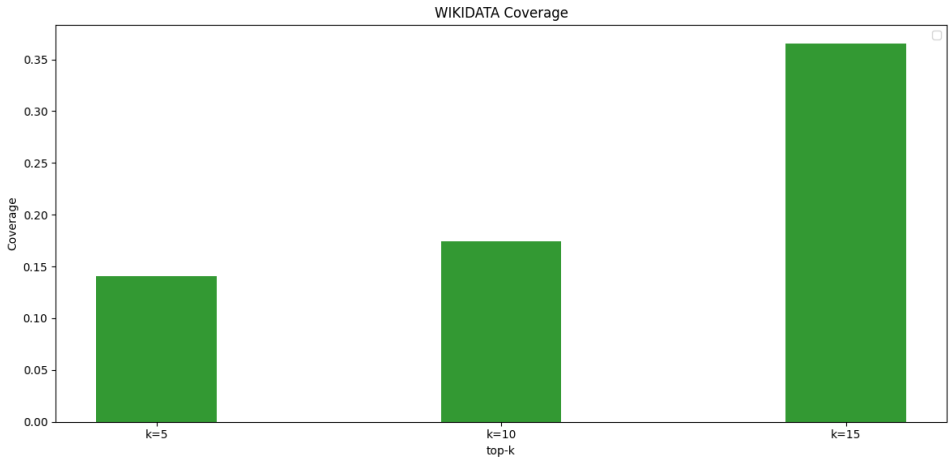

Figure 5.5: Coverage for various k in unseen query workload.

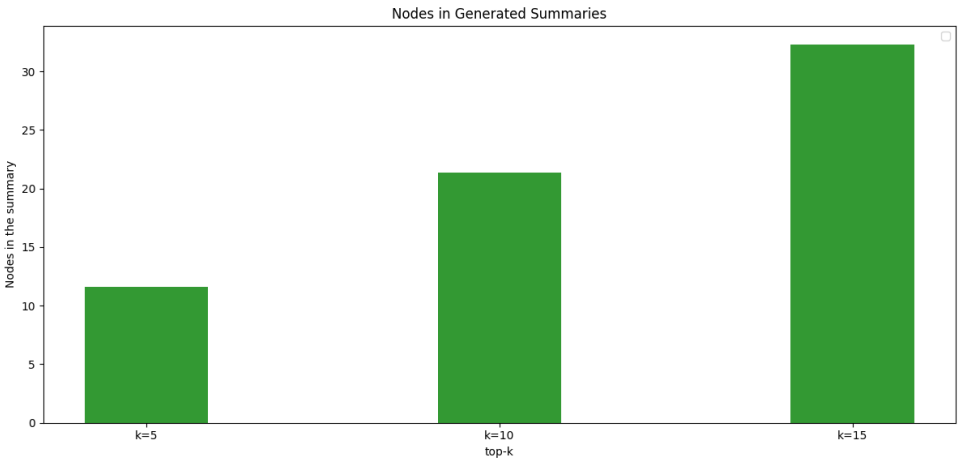

Figure 5.6: Nodes in summary for various k.

requires for calculating the corresponding summary which is in line with the complexity of the presented summary. It requires at the worst case almost four minutes for calculating the summary for k=15, which we would say is an acceptable performance for searching and calculating shortest paths through more than 10k queries.

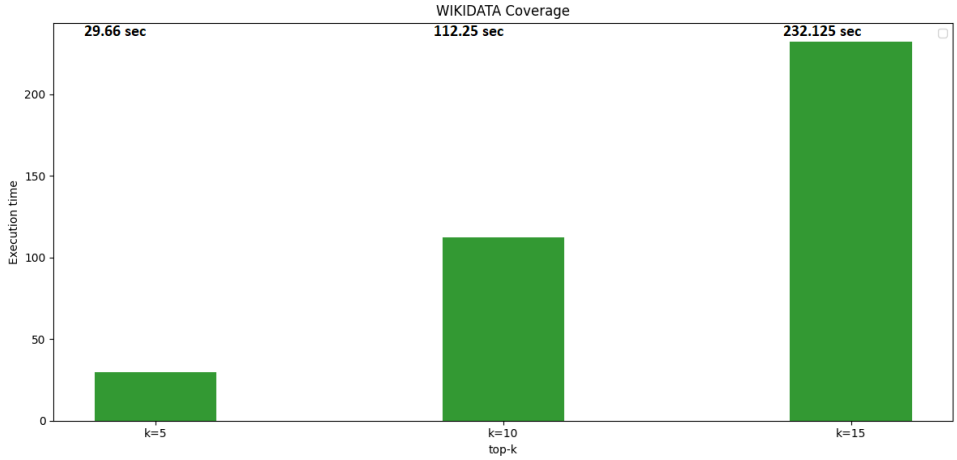

Figure 5.7: Execution time for calculating summaries for the various k

Chapter 6

Conclusions

In this thesis we present a summarization method able to construct personalized, query driven, semantic summaries with high-quality. More specifically, based on user input we first select the most important nodes that usually appear in the query workload along with user input, and then we also exploit user queries in order to connect user provided input with the most important nodes identified. To achieve that we find the most frequent minimal paths from user queries and we exploit them for connecting the selected nodes. We performed an extensive evaluation on two real datasets, i.e. WikiData and DBPedia demonstrating the high quality of the generated summary, by maximizing coverage and as such providing ideal summaries for user exploration.

Future work can exploit alternative methods for linking the nodes in the top_k and user input, minimizing the number of times that the original graph should be queried, or by exploring also connecting one node at a time with the so far constructed summary, trying to minimize further the additional nodes introduced. Experiments on other datasets could also be used to further generalize the results of this thesis.

Bibliography

[1] Anas Alzogbi and Georg Lausen. Similar structures inside rdf-graphs. In *LDOW*, 2013.

[2] Burton H Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13(7):422–426, 1970.

[3] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. Tractable reasoning and efficient query answering in description logics: The dl-lite family. *Journal of Automated reasoning*, 39(3):385–429, 2007.

[4] Sejla Cebiric, François Goasdoué, Haridimos Kondylakis, Dimitris Kotzinos, Ioana Manolescu, Georgia Troullinou, and Mussab Zneika. Summarizing semantic graphs: a survey. *VLDB J.*, 28(3):295–327, 2019.

[5] Stanislav Malyshev, Markus Krötzsch, Larry González, Julius Gonsior, and Adrian Bielefeldt. Getting the most out of wikidata: Semantic technology usage in wikipedia's knowledge graph. In Denny Vrandecic, Kalina Bontcheva, Mari Carmen Suárez-Figueroa, Valentina Presutti, Irene Celino, Marta Sabou, Lucie-Aimée Kaffee, and Elena Simperl, editors, *The Semantic Web - ISWC 2018 - 17th International Semantic Web Conference, Monterey, CA, USA, October 8-12, 2018, Proceedings, Part II*, volume 11137 of *Lecture Notes in Computer Science*, pages 376–394. Springer, 2018.

[6] Enrico Motta, Paul Mulholland, Silvio Peroni, Mathieu d'Aquin, Jose Manuel Gomez-Perez, Victor Mendez, and Fouad Zablith. A novel approach to visualizing and navigating ontologies. In *International Semantic Web Conference*, pages 470–486. Springer, 2011.

[7] Silvio Peroni, Enrico Motta, and Mathieu d'Aquin. Identifying key concepts in an ontology, through the integration of cognitive principles with statistical and topological measures. In *Asian Semantic Web Conference*, pages 242–256. Springer, 2008.

[8] Georgia Eirini Trouli, Georgia Troullinou, Lefteris Koumakis, Nikolaos Papadakis, and Haridimos Kondylakis. Summer: Summarizing RDF/S kbs using machine learning. In Oshani Seneviratne, Catia Pesquita, Juan Sequeda, and Lorena Etcheverry, editors, *Proceedings of the ISWC 2021 Posters, Demos and Industry Tracks: From Novel Ideas to Industrial Practice co-located with 20th International Semantic Web Conference (ISWC 2021), Virtual Conference, October 24-28, 2021*, volume 2980 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2021.

[9] Georgia Troullinou, Haridimos Kondylakis, Kostas Stefanidis, and Dimitris Plexousakis. Exploring RDFS kbs using summaries. In Denny Vrandecic, Kalina Bontcheva, Mari Carmen Suárez-Figueroa, Valentina Presutti, Irene Celino, Marta Sabou, Lucie-Aimée Kaffee, and Elena Simperl, editors, *The Semantic Web - ISWC 2018 - 17th International Semantic Web Conference, Monterey, CA, USA, October 8-12, 2018, Proceedings, Part I*, volume 11136 of *Lecture Notes in Computer Science*, pages 268–284. Springer, 2018.

[10] Giannis Vassiliou, Georgia Troullinou, Nikolaos Papadakis, and Haridimos Kondylakis. Wbsum: Workload-based summaries for RDF/S kbs. In Qiang Zhu, Xingquan Zhu, Yicheng Tu, Zichen Xu, and Anand Kumar, editors, *SSDBM 2021: 33rd International Conference on Scientific and Statistical Database Management, Tampa, FL, USA, July 6-7, 2021*, pages 248–252. ACM, 2021.

[11] Giannis Vassiliou, Georgia Troullinou, Nikos Papadakis, Kostas Stefanidis, Evangelia Pitoura, and Haridimos Kondylakis. Coverage-based summaries for RDF kbs. In Ruben Verborgh, Anastasia Dimou, Aidan Hogan, Claudia d'Amato, Ilaria Tiddi, Arne Bröring, Simon Maier, Femke Ongenae, Riccardo Tommasini, and Mehwish Alam, editors, *The Semantic Web: ESWC 2021 Satellite Events - Virtual Event, June 6-10, 2021, Revised Selected Papers*, volume 12739 of *Lecture Notes in Computer Science*, pages 98–102. Springer, 2021.

[12] Gang Wu, Juanzi Li, Ling Feng, and Kehong Wang. Identifying potentially important concepts and relations in an ontology. In *International Semantic Web Conference*, pages 33–49. Springer, 2008.

[13] Xiang Zhang, Gong Cheng, and Yuzhong Qu. Ontology summarization based on rdf sentence graph. In *Proceedings of the 16th international conference on World Wide Web*, pages 707–716, 2007.