

# Quality Metrics For RDF Graph Summarization

**Editor(s):** Name Surname, University, Country

**Solicited review(s):** Name Surname, University, Country

**Open review(s):** Name Surname, University, Country

Mussab Zneika<sup>a</sup> Dan Vodislav<sup>a</sup> and Dimitris Kotzinos<sup>a</sup>

<sup>a</sup> *ETIS, UMR 8051 Universite Paris Seine, Universite Cergy-Pontoise, ENSEA, CNRS*

*E-mail: Mussab.Zneika@ensea.fr; Dan.Vodislav@u-cergy.fr; Dimitrios.Kotzinos@u-cergy.fr*

**Abstract.** RDF Graph Summarization pertains to the process of extracting concise but meaningful summaries from RDF Knowledge Bases (KBs) representing as close as possible the actual contents of the KB. RDF Summarization allows for better exploration and visualization of the underlying RDF graphs, optimization of queries or query evaluation in multiple steps, better understanding of connections in Linked Datasets and many other applications. In the literature, there are efforts reported presenting algorithms for extracting summaries from RDF KBs. These efforts though provide different results while applied on the same KB, thus a way to compare the produced summaries and decide on their quality, in the form of a quality framework, is necessary. So in this work, we propose a comprehensive Quality Framework for RDF Graph Summarization that would allow a better, deeper and more complete understanding of the quality of the different summaries and facilitate their comparison. We work at two levels: the level of the ideal summary (or ideal schema) of the KB that could be provided by an expert user and the level of the instances contained by the KB. For the first level, we are computing how close the proposed summary is to the ideal solution (when this is available) by computing its precision and recall against the ideal solution. For the second level, we are computing if the existing instances are covered (i.e. can be retrieved) and in what degree by the proposed summary. We use our quality framework to test the results of three of the best RDF Graph Summarization algorithms, when summarizing different (in terms of content) and diverse (in terms of total size and number of instances, classes and predicates) KBs and we present comparative results for them. We conclude this work by discussing these results and the suitability of the proposed quality framework in order to get useful insights for the quality of the presented results.

**Keywords:** Quality framework; Quality metrics; RDF Summarization; Linked Open Data; RDF Query processing

## 1. Introduction

RDF has become one of the major standards in describing and publishing data, establishing what we call the Semantic Web. Thus, the amount of RDF data available increases fast both in size and complexity, making the appearance of RDF Knowledge Bases (KBs) with millions or even billions of triples something usual. Given that RDF is built on the promise of linking together relevant datasets or KBs and with the appearance of the Linked Open Data (LOD) cloud, we can now query KBs (both standalone or distributed) with millions or billions of triples altogether. This in-

creased size and complexity of RDF KBs has a direct impact on the evaluation of the RDF queries we express against these RDF KBs. Especially on the LOD cloud, we observe that a query against a big, complex, interlinked and distributed RDF KB might retrieve no results at the end because either the association between the different RDF KBs is weak (is based only on a few associative links) or there is an association at the schema level that has never been instantiated at the actual data level. Moreover, a lot of these RDF KBs carry none at all or only partial schema information (mainly contain instances build and described separately). Additionally, in the LOD cloud the number

of KBs which do not use the full schema or they use multiple schemas is increased due to the absence of the schema information which describes the interlinks between the datasets and the combinatorial way of mixing vocabularies.

One way to address the concerns described above is by creating summaries of the RDF KBs. Thus we allow the user or the system to decide whether or not to post a query, since she knows whether information is present or not based on the summary. This would provide significant cost savings in processing time since we will substitute queries on complex RDF KBs with queries first on the summaries (on much simpler structures with no instances) and then with queries only towards the KBs that we know will produce some useful results. Graph summarization techniques would allow the creation of a concise representation of the KB regardless of the existence or not of schema information in the KB. Actually, the summary will represent the actual situation in the KB, namely should capture the *existing/used* classes and relationships by the instances and not what the schema proposes (and might have never been used). This should facilitate the query building for the end users with the additional benefit of exploring the contents of the KB based on the summary. This is true regardless if we use heterogeneous or homogeneous, linked or not, standalone or distributed KBs. In all these cases we can use the RDF summary to concisely describe the data in the RDF KB and possibly add useful information for the RDG graph queries, like the distribution and the number of instances for each involved entity.

In the literature we can find various efforts proposing summarization techniques for RDF graphs. These techniques, presented briefly in section 3, come from various scientific backgrounds ranging from generic graph summarization to explicit RDF graph summarization. While all promise that they provide correct, concise and well-built summaries so far has been very little effort into address in a comprehensive and coherent way the problem of evaluating these summaries against different criteria and have some mathematical metrics to describe the quality of the results. Only sparse efforts have been reported, usually tailored to a specific method or algorithm. So with this paper, we aim to cover the gap that exists in the literature and provide a comprehensive Quality Framework for RDF Graph Summarization that would allow a better, deeper and more complete understanding of the quality of the different summaries and facilitate their comparison. We propose to take into account the possibil-

ity to compare the summary against two levels of information possibly available for a RDF KB. In the case where an ideal summary exists, either because it has been proposed by a human expert or because we can assume that an existing schema represents perfectly the data graph, we compare the summary provided by the algorithms with it and use similarity measures to compute its precision and recall against the ideal summary. If this is not available or additionally to it, we compute the percentage of the instances represented by the summary (including both class and property instances). This provides us with the understanding of how well the summary covers the KB. One can combine at the end the two overall metrics or use them independently. In order to validate the proposed quality metrics, we evaluated three of the most promising PDF graph summarization algorithms and report on the quality of their results over different datasets with diverse characteristics. We should note here that the proposed Quality Framework is independent of any of the algorithms evaluated but it is suitable in providing a common ground to compare them.

The paper is structured as follows: Section 2 introduces some of the foundations of RDF and RDFS, which are useful for defining later on some concepts in our work; Section 3 provides a review of the existing works around quality metrics in graph summarization; while Section 4 presents our proposed Quality Metrics for RDF Graph Summaries. Section 5 presents the three of the most promising RDF Graph Summarization algorithms in the literature that are compared using the proposed Quality Framework in Section 6, where the extensive experiments performed in order to validate the appropriateness of the proposed metrics are reported. We then conclude our paper in section 7.

## 2. Preliminaries

As per the W3C standards, the RDF data model represents data on the Web as a set of triples of the form  $(s, p, o)$ , expressing the fact that for the subject  $s$ , the value of the property  $p$  is the object/value  $o$ . RDF data can also be represented as a labeled directed graph in which entities (subjects/objects) are represented as nodes and property instances (expressed by the triples) as labeled directed edges. RDF datasets are usually accompanied with a RDF Schema<sup>1</sup>, which pro-

---

<sup>1</sup><https://www.w3.org/TR/2004/REC-rdf-schema-20040210/>

Let  $C, P, I$  and  $L$  be the sets of *class* Universal Resource Identifiers (URIs), *property* URIs, *instance* URIs and *literal* values respectively, and let  $T$  be a set of RDFS standard properties (*rdfs:range*, *rdfs:domain*, *rdf:type*, *rdfs:subClassOf*, etc.). The concepts of RDF schemas and instances can be formalized as follows.

- $N_s$  is the set of nodes, representing classes and properties.
- $E_s \subseteq \{(x, \alpha, y) \mid x \in N_s, \alpha \in T, y \in N_s\}$  is the set of labeled edges.
- $\lambda_s : N_s \longrightarrow C \cup P$  is an injective node labeling function that maps nodes of  $N_s$  to class and property URIs.

**Definition 2 (RDF data graph).** An RDF data graph  $G_i = (N_i, E_i, \lambda_i, I, P, L)$  is a directed labeled graph where:

- We note  $\lambda_{ei} : E_i \longrightarrow P$  the edge labeling function that associates to each edge  $(x, \alpha, y) \in E_i$  the property URI  $\alpha \in P$ .

The diagram illustrates the structure of the dataset, showing two main parts separated by a dashed line. The top part shows a network of entities and their relationships. The bottom part shows a sequence of entities and their relationships, with a 'Roll type' label at the bottom right.

**Top Part (Network Structure):**

- Entities:** String, Painter, Painting, Museum.
- Relationships:**
  - String to Painter: frame, name
  - Painter to Painting: paints
  - String to Painting: title, creator, technique
  - Painting to Museum: exhibited

**Bottom Part (Sequence Structure):**

- Entities:** Pablo, Picasso, Rembrandt, Woman, Guernica, Abraham, museum.es.
- Relationships:**
  - Pablo to Picasso: frame, name
  - Pablo to Rembrandt: frame, name
  - Pablo to Woman: frame, name
  - Pablo to Guernica: frame, name
  - Pablo to Abraham: frame, name
  - Pablo to museum.es: frame, name
  - Pablo to Picasso: paints
  - Pablo to Rembrandt: paints
  - Pablo to Woman: paints
  - Pablo to Guernica: paints
  - Pablo to Abraham: paints
  - Pablo to museum.es: paints
  - Woman to Guernica: exhibited
  - Guernica to Abraham: exhibited
  - Abraham to museum.es: exhibited

**Roll type** is indicated at the bottom right.

Fig. 1.: RDF Schema and data graphs

**Type edges.** Edges labeled with *rdf:type* in the RDF data graph explicitly describe the type (class) of an instance, e.g. dashed edges in Fig. 1, where for instance *Picasso* is declared to be a *Painter*. We will note in the following the type edge label with  $\tau$ . For an instance  $x \in N_i$ , we define  $Types(x) = \{y \mid (x, \tau, y) \in E_i\}$  to be the set of types related to the node  $x$  via an explicit type edge definition, e.g.,  $Types(Picasso) = \{Painter\}$ , while  $Types(Guernica) = \{Painting\}$ .

**Attributes.** We denote by  $Attributes(x) = \{\alpha : \forall (x, \alpha, y) \in E_i : \alpha \neq \tau \wedge \lambda_i(y) \in L \wedge x \in N_i\}$  a set of labels of the non-Type edges which associate the node  $x$  with a set of literal nodes (nodes labeled by *literal* values),

Bisimilarity in a directed labeled graph is an Equivalence Relation defined on a set of nodes  $N$ , such that two nodes  $(u,v)$  are Bisimilar if and only if the set of edges coming immediately out of  $u$  is equal to the set of edges coming immediately out of  $v$  and also, all successor nodes of  $u$  and  $v$  must be Bisimilar( in other words, the outgoing paths of  $u$  and  $v$  are similar ). We call the Bisimilarity relation when defined based on

outgoing paths, Forward (FW) Bisimilarity, and when it is based on incoming paths, Backward (BW) Bisimilarity.

**Knowledge pattern.** A knowledge pattern (or simply pattern from now on) characterizes a set of instances in an RDF data graph that share a common set of types and a common set of properties. More precisely, a knowledge pattern  $KP$  in an RDF data graph is a quad  $(Cl, Pr, Ins, SUP)$ , where  $Cl = \{c_1, c_2, \dots, c_n\} \subseteq C$  is a set of classes,  $Pr = \{Pr_1, Pr_2, \dots, Pr_m\} \subseteq P$  is a set of properties,  $Ins \subseteq I$  is the set of instances that have all the types of  $Cl$  and all the properties of  $Pr$ , and  $SUP = |Ins|$  is called *the support* of the knowledge pattern in the RDF data graph. For instance,  $Ins$  corresponds to the set of bindings for the  $?a$  variable over the RDF data graph in the following SPARQL-like conjunctive pattern:  $\{<?a, \tau, c_1 >, <?a, \tau, c_2 >, \dots, <?a, \tau, c_n >, <?a, Pr_1, ?b_1 >, <?a, Pr_2, ?b_2 >, \dots, <?a, Pr_m, ?b_m >\}$ . We introduce the term knowledge pattern because it is not sure that all summarisation algorithms will produce something that can be necessarily defined as an RDF class and also because we want to differentiate from the classes of the ideal summary when we compare the two.

### 3. Related work

RDF graph summarization has been intensively studied, with various approaches and techniques proposed to summarize the RDF graphs, which could be grouped into four main categories:

1. *Aggregation and grouping approaches* [27,28, 29? ,32,23], which are based on grouping the nodes of an input RDF graph  $G$  into clusters/groups based on the similarity of the attribute values and on the neighborhood relationships associated with nodes of  $G$ .
2. *Structural extraction approaches* [12,20,13,19, 31,17,18,9,21,22,26], which define an equivalence relation on the nodes of the RDF data graph  $G$ , usually based on the set of incident graph paths. This allows extracting a form of schema for  $G$  by representing the equivalence classes of nodes of  $G$  as nodes in the summary graph, characterized by the set of incident paths of each class.
3. *Logical compression approaches* [15,16], which are based on compressing the RDF datasets by generating a set of logical rules from the dataset

and removing triples that can be inferred from these rules. The summary graph is then represented by a compressed graph and set of logical decompression rules, with the drawback that such approaches do not produce RDF graphs as summaries.

4. *Pattern-mining-based approaches* [33,14,34], which are based on extracting frequent patterns from the RDF graph, then composing them to build an approximated summary graph.

Typically, the RDF summarization methods proposed so far do not address in depth the problem of the quality of the produced RDF summaries. A noticeable exception is the work in [11], which proposes a model for evaluating the precision of the graph summary, compared to a gold standard summary. The main idea of the precision model is based on counting the edges or paths that exist in the summary and/or in the data graph. The precision of a summary is evaluated in the standard way, based on the number of true positives (the number of edges existing in the summary and in the input graph) and false positives (the number of invalid edges and paths existing in the summary but not in the input graph). The limitation of this quality model is that the precision alone cannot accurately assess the quality, since a high precision can be achieved at the expenses of a poor recall by returning only few (even if correct) common paths.

Besides that, only few efforts have been reported in the literature addressing the quality of the schema summarization methods in general [30,25,10], i.e. the quality of the RDF schema that can be obtained through RDF summarization. The quality of the RDF schema summary in [25] is based on expert ground truth and is calculated as the ratio of the number of classes identified both by the expert users and the summarization tool over the total number of classes in the summary. The main limitation of this approach is that it uses a Boolean match of classes and fails to take into account similarity between classes when classes are close but not exactly the same as in the ground truth or when classes are represented by more than one class in the summary. Works in schema matching (e.g. [30]) are also using to some extent similar metrics like recall, precision, F1-Measure commonly used in Information Retrieval, but are not relevant to our work since even if we consider an RDF graph summary as an RDF schema, we are not interested in matching its classes and properties one by one, since as stated above this binary view of the summary results does not offer much

in the quality discussion. Additionally these works do not take into account issues like the size of the summary.

To the best of our knowledge, this is the first effort in the literature to provide a comprehensive Quality Framework for RDF Graph Summarization, independent of the type and specific results of the algorithms used and the size, type and content of the KBs. We provide metrics that help us understand not only if this is a valid summary but also if a summary is better than another in terms of the specified quality characteristics. And we can do this by assessing information, if available, both at schema and instance levels.

#### 4. Quality Assessment Model

In this section we present a comprehensive and coherent way to measure the quality of RDF summaries produced by any algorithm that summarises RDF graphs. The framework is independent of the way algorithms work and makes no assumptions on the type or structure neither of the input nor of the final results, besides being expressed in RDF; this is required in order to guarantee the validity of the result but can be easily extended to other cases of semantic summarisation, like for graphs expressed in OWL or Description Logics. In order to achieve this, we work at two levels:

- *schema level*, where if an ideal summary exists, the summary is compared with it by computing the precision and recall for *each* class and its neighbourhood (properties and attributes having as domain that class) of the produced summary against the ideal one; we also compute the precision and recall of the whole summary against the ideal one. The first will capture the quality of the summary at the local (class) level, while the second will give us the overall quality in terms of classes' and properties/attributes' precision and recall.
- *instance level*, where the coverage that the summary provides for class and property instances is calculated, i.e. how many instances will be retrieved if we query the whole summary graph. We use again precision and recall against the contents of the original KB.

At the end a metric is presented that provides an indication of the quality of the graph summary by measuring whether or not the summary is a connected

graph. Ideally, a summary should be a connected graph but this also depends on the actual data stored in the Knowledge Base. Thus a disconnected graph could be an indication of the data quality in the KB and not necessarily a problem of the summarisation process. Nevertheless, we present it here as another indicator of the quality process, especially if the summary is compared with an ideal one, but for the reason mentioned before we avoid to combine it with the rest of the presented metrics. Finally, we discuss some results that combine these metrics and interpret their meaning.

##### 4.1. Quality Metrics in the presence of an ideal summary

In this section we present our quality assessment framework to evaluate the quality of an RDF graph summary against a ground truth summary (**S**) (e.g. one provided by an expert). We measure how close the proposed summary is to the ground truth summary by computing its precision and recall against this ground truth. We suggest that we compute both the precision and recall at the class and at the property level and at the overall summary level. Table 1 gives us a summary Description of the schema-level proposed measures.

**Precision and Recall for classes** We present here the recall and the precision measures for the classes of the detected patterns against a ground truth summary *S*. We first introduce the recall over the classes which is the fraction of relevant classes that are reported in the summary. Given a set of knowledge patterns  $\Pi$  (as defined in Section 2 and referred commonly as patterns from now on) and a set of classes  $C \in S$ , we start by defining the recall of a class  $c \in C$  over the set of patterns  $\Pi$  as the fraction of relevant class's properties (namely properties that have this class as their domain) that are reported in  $\Pi$ , we denote it by schema class recall  $SchemaRec(c, \Pi)$ :

$$SchemaRec(c, \Pi) = \frac{|\bigcup_{pa \in \Pi} (A(c) \cap A(pa))|}{|A(c)|} \quad (1)$$

The  $A(pa)$  is the set of properties and attributes involved in the pattern  $pa$ , and the  $A(c)$  is the set of properties and attributes of the ideal class  $c$ . Thus, the overall summary recall using the classes  $SchemaRec_{ClassAll}$  is computed as the mean of the various schema class recall  $SchemaRec(c, \Pi)$  for all the classes  $c$  of the

Measure	What it indicates	How it is computed
$SchemaRecall(c, \Pi)$	Schema recall of a class $c$ over the set of patterns $\Pi$ .	Divide the number of relevant class's properties that are reported in $\Pi$ on the total number class's properties.
$SchemaRec_{ClassAll}$	Overall schema class recall.	Compute the mean of the various $SchemaRecall(c, \Pi)$ for all the classes $c$ of the ground-truth Schema $S$ .
$Sim(pa, c)$	Similarity between a class $c$ and a pattern $pa$ .	Divide the number of common properties between the class $c$ and the pattern $pa$ on the total number of $pa$ properties $pa$ .
$Nps(c)$	The number of patterns that represent the class $c$	Count all the patterns having $Sim(pa, c) > 0$ .
$SchemaPrec(c, \Pi)$	Schema class precision of the class $c$ over the set of patterns $\Pi$ .	Sum the $sim(pa, c)$ for all the patterns of $\Pi$ .
$SchemaPrec_{ClassAll}$	Overall schema class precision.	Compute the mean of the various class precision values $SchemaPrec(c, \Pi)$ for all the retrieved classes of the ground-truth Schema $S$ .
$SchemaF1_c$	Schema class F-Measure.	Combine the $SchemaPrec_{ClassAll}$ and $SchemaRec_{ClassAll}$ using the standard formula of the F-Measure.
$SchemaRec_{PropertyAll}$	Overall Schema property recall.	Divide the number of relevant properties extracted by the summary on the total number of properties in the ground truth schema.
$SchemaF1_p$	Schema property F-Measure.	Combine the $SchemaPrec_{PropertyAll}$ and $SchemaRec_{PropertyAll}$ using the standard formula of the F-Measure
$SchemaF1$	Overall schema F-measure.	Combine the class schema F-Measure $SchemaF1_c$ and property schema F-Measure $SchemaF1_p$ .

Table 1: Summary description of the proposed Schema Measures

ground-truth Schema  $S$ .

$$SchemaRec_{ClassAll} = \frac{1}{|C|} \sum_{c \in C} SchemaRecall(c, \Pi) \quad (2)$$

The precision is the fraction of retrieved classes and properties of the summary that are relevant. If a knowledge pattern of a summary carries a *typeof* link then this pattern is relevant to a specific class if the *typeof* points to this class, if not this is not relevant to this class. If no *typeof* information exists then we use the available properties and attributes to evaluate the similarity between a class and a pattern. Thus we define the  $L(c, pa)$  function to capture this information and we add this to the similarity function.

$$L(c, pa) = \begin{cases} 1, & \text{if } typeof(pa) = c \text{ or } typeof(pa) = \emptyset \\ 0, & \text{otherwise} \end{cases} \quad (3)$$

The similarity between a class  $c$  in the ideal summary and a pattern  $pa$   $Sim(pa, c)$  in the computed summary is defined as the number of common properties between class  $c$  and pattern  $pa$  divided on the total number of the properties of the patterns  $pa$ :

$$Sim(pa, c) = L(pa, c) * \frac{|(A(c) \cap A(pa))|}{|A(pa)|} \quad (4)$$

Given that a class might be represented by more than one knowledge patterns, depending on the algorithm used, we are interested in introducing a way to pe-

nalise cases where this happens, thus favouring smaller summaries over bigger ones. We achieve this by introducing a weight function that allows us to reduce the similarity value if this is based on consuming multiple patterns. Thus we introduce the following exponential function, which uses coefficient  $a$  to allow variations if needed in the future, and is chosen based on experimental evaluation of the functions that could provide us a smooth decay in similarity as patterns' number increases. The  $Nps(c)$  is the number of patterns that represent the class  $c$  and  $\alpha \in [1, 10]$ .

$$W(c) = e^{1 - \frac{a}{\sqrt{Nps(c)}}} \quad (5)$$

Based on this weight function we define the class precision metric for every pattern  $pa$  in the computed summary and every class  $c$  in the ground truth summary as follows:

$$SchemaPrec(c, \Pi) = W(c) * \frac{\sum_{pa \in \Pi} Sim(pa, c)}{Nps(c)} \quad (6)$$

Thus, we define the schema class precision  $SchemaPrec_{ClassAll}$  as the mean of the various class precision values  $SchemaPrec(c, \Pi)$  for all the classes of the ground-truth Schema  $S$ .

$$SchemaPrec_{ClassAll} = \frac{\sum_{c \in C} SchemaPrec(c, \Pi)}{|C1|} \quad (7)$$

where  $C1 \in C$  is the list of all the ground truth's retrieved classes, or in other words, is the list of the

ground truth's classes for which  $SchemaPrec(c, \Pi) > 0$ .

However, neither precision nor recall alone can accurately assess the match quality. In particular, recall can easily be maximized at the expense of a poor precision by returning as many correspondences as possible. On the other side, a high precision can be achieved at the expense of a poor recall by returning only few (correct) correspondences. Hence it is necessary to consider both measures and express this through a combined measure; we use the F-Measure for this purpose, namely  $SchemaF1_c$ :

$$SchemaF1_c = 2 * \frac{SchemaPrec_{ClassAll} * SchemaRec_{ClassAll}}{SchemaPrec_{ClassAll} + SchemaRec_{ClassAll}} \quad (8)$$

**Precision and Recall for properties** The overall recall at the property level, namely  $SchemaRec_{PropertyAll}$  is computed as the ratio between the total number of common properties extracted by the summary and the total number of properties in the ground truth summary:

$$SchemaRec_{PropertyAll} = \frac{|\bigcup_{pa \in \Pi} A(pa) \cap \bigcup_{c \in C} A(c)|}{|\bigcup_{c \in C} A(c)|} \quad (9)$$

Since there is no false positive for properties because the algorithms do not invent new properties, the  $\bigcup_{pa \in \Pi} A(pa) \cap \bigcup_{c \in C} A(c) = \bigcup_{pa \in \Pi} A(pa)$ . Thus, We can re-formulate the previous equation as follows:

$$SchemaRec_{PropertyAll} = \frac{|\bigcup_{pa \in \Pi} A(pa)|}{|\bigcup_{c \in C} A(c)|} \quad (10)$$

We note that the schema precision at the property level in our experiments is always equal to 1, since there are no false positives for properties because the way summarisation algorithms work, they do not invent new properties, at worst they might miss some. Thus the F-Measure for the schema properties, namely  $SchemaF1_p$  will depend only on the  $SchemaRec_{PropertyAll}$ :

$$SchemaF1_p = 2 * \frac{SchemaPrec_{PropertyAll} * SchemaRec_{PropertyAll}}{SchemaPrec_{PropertyAll} + SchemaRec_{PropertyAll}} \quad (11)$$

By defining the individuals metrics for the class schema F-Measure  $SchemaF1_c$  and property schema F-Measure, we can define the combined overall schema F-measure  $SchemaF1$  as the weighted harmonic mean of the schema F-Measure and property schema F-Measure :

$$SchemaF1 = \beta * SchemaF1_p + (1 - \beta) * SchemaF1_c \quad (12)$$

where the weight  $\beta \in [0, 1]$ . The overall schema F-measure can be viewed as a compromise between class schema F-Measure and property schema F-Measure. It is high only when both class and property schema F-Measure are high. It is equivalent to class schema F-Measure when  $\beta = 0$  and to property schema F-Measure when  $\beta = 1$ . But it is not adequate to provide by itself a full picture of the summarisation process; thus the readers should also refer to the rest of the metrics, too, when they want to get a better idea on the quality of the computed summary.

#### 4.2. Quality Model At Instance Level

We measure the quality with regard to the instances by introducing the notion of the coverage of the instances of the original KB, i.e. how many of the original class and property instances are successfully represented by the computed RDF summary graph (e.g. can be retrieved in the case of a SPARQL query). This requires computing both the precision and recall at the class instance and at the property instance levels. The Table 2 gives us a summary description of the proposed instance measures.

**Precision and Recall for class instances** The overall recall at the instance class level is the total number of the class instances represented by the computed summary divided on the total number of instances of the original KB D.

$$InstanceRec_{ClassAll} = \frac{|instances(\Pi)|}{|instances(D)|} \quad (13)$$

The class  $instances(\Pi)$  is the list of instances covered by the set of patterns  $\Pi$ ,  $instances(D)$  is the list of all instances of the original KB D. To avoid the problem of overlapping of instances in several patterns which will cause the over-coverage, we calculate the  $instances(\Pi)$ ,  $instances(D)$  as follows:

$$instances(\Pi) = \bigcup_{pa \in \Pi} instances(pa) \quad (14)$$

Measure	What it indicates	How it is computed
$instances(c)$	The list of class $c$ instances.	—
$instances(p)$	The list of subjects which have the property $p$ .	—
$instances(pa)$	The list of covered class instances by the pattern $pa$ .	—
$instances(\Pi)$	The list of class instances covered by the set of patterns $\Pi$ .	—
$instances(D)$	The list of all class instances of original KB $D$ .	—
$Cov_c(c, pa)$	The list of the class instances which are represented by a pattern $pa$ .	Get the $instances(pa)$ if the pattern $pa$ is relevant to the class $c$ or $\emptyset$ otherwise.
$instances(c, \Pi)$	The total number of class instances that are reported by a set of patterns $\Pi$ representing the class $c$ .	Sum the $ Cov_c(c, pa) $ for all the patterns of the $\Pi$ .
$InstancePrec(c, \Pi)$	The instance class precision of a class $c$ over the set of patterns $\Pi$ .	Divide the number of original instances of the class $c$ reported in $\Pi$ on $instances(c, \Pi)$ .
$InstancePrec_{ClassAll}$	Overall instance class precision.	The mean of the various $InstancePrec(c, \Pi)$ for all the classes of the ground-truth Schema $S$ .
$InstanceF1_c$	Instance class F-Measure.	Combine the $InstancePrec_{ClassAll}$ and $SchemaRec_{ClassAll}$ using the standard formula of the F-Measure.
$Cov_p(p, pa)$	The list of the original property instances which are successfully represented by a pattern $pa$ .	Get the $instances(p)$ if the property $p$ is reported in the pattern $pa$ or get $\emptyset$ otherwise.
$instances(p, \Pi)$	The list of the original property $p$ instances that are successfully covered by a set of patterns $\Pi$ .	The Union of the $Cov_p(p, pa)$ for all the $\Pi$ .
$InstanceRec(p, \Pi)$	The instance property recall.	Divide $ \Pi instances(p, \Pi) $ on $instances(p)$ .
$InstanceRec_{PropertyAll}$	Overall recall at the instance property level	Weighted mean of the various $InstanceRec(p, \Pi)$ for all the properties of the ground-truth.
$InstancePrec(p, \Pi)$	The precision of a property $p$ in $P$ over the set of patterns $\Pi$ .	
$InstancePrec_{PropertyAll}$	Overall instance property precision	Mean of the various $InstanceRec(p, \Pi)$ for all the covered properties of the ground-truth.
$InstanceF1_p$	Instance property F-Measure	Combine the $InstancePrec_{PropertyAll}$ and $InstanceRec_{PropertyAll}$ using the standard formula of the F-Measure.
$InstanceF1$	Overall instance F-measure .	Combine the class Instance F-Measure $InstanceF1_c$ and property Instance F-Measure $InstanceF1_p$ .

Table 2: Summary Description of the proposed Instance Measures

$$instances(D) = \bigcup_{c \in C} instances(c) \quad (15) \quad \text{low:}$$

The  $instances(pa)$  denotes the list of covered instances by the pattern  $pa$  and the  $instances(c)$  denotes the list of instances of the type  $c$  in the original KB  $D$ .

We denote by  $Cov_c(c, pa)$ , the list of the class instances which are represented by a pattern  $pa$ :

$$Cov_c(c, pa) = \begin{cases} instances(pa), & \text{if } L(c, pa) = 1 \\ \emptyset, & \text{otherwise} \end{cases} \quad (16)$$

$InstancePrec(p, \Pi)$ , the precision of a property  $p$  in  $P$  over the set of patterns  $\Pi$  and then in the case of multiple patterns representing the class instances:

$$instances(c, \Pi) = \sum_{pa \in \Pi} |Cov_c(c, pa)| \quad (17)$$

where the  $instances(c, \Pi)$  is the total number of class instances that are reported by a set of patterns  $\Pi$  representing the class  $c$ .

We define  $InstancePrec(c, \Pi)$  the instance precision of a class  $c$  in  $C$  over the set of patterns  $\Pi$  as fol-

$$InstancePrec(c, \Pi) = \frac{|instances(c) \cap instances(c, \Pi)|}{|instances(c, \Pi)|} \quad (18)$$

Thus, we define the overall instance class precision denoted by  $InstancePrec_{ClassAll}$  as the weighted mean of the various  $InstancePrec(c, \Pi)$  for all the retrieved classes:

$$InstancePrec_{ClassAll} = \frac{\sum_{c \in C} wi(c) * InstancePrec(c, \Pi)}{|C1|} \quad (19)$$

The  $wi(c)$  is the weight of a class  $c$  and it measures the percentage of class instances of the class  $c$  with respect to the total number of class instances in the KB. This is used to wight in the importance of the specific class in terms of the number of instances it "represents"; so the more instances it "represents" the bigger the weight. It is defined as the number of instances of class  $c$  in the KB  $instances(c)$  compared to the total number of



property instances in the KB  $instances(D)$ .

$$wi(c) = \frac{instances(c)}{instance(D)} \quad (20)$$

The overall instance class recall and the overall instance class precision are combined by the instance class F-Measure, namely  $InstanceF1_c$ :

$$InstanceF1_c = 2 * \frac{InstancePrec_{ClassAll} * InstanceRec_{ClassAll}}{InstancePrec_{ClassAll} + InstanceRec_{ClassAll}} \quad (21)$$

**Precision and Recall at Property Level** The  $Cov(p, pa)$  represents the list of the original property instances which are successfully represented by a pattern  $pa$ :

$$Cov_p(p, pa) = \begin{cases} instances(pa), & \text{if } p \in pa \\ \emptyset, & \text{otherwise} \end{cases} \quad (22)$$

We denote by the  $instances(p, \Pi)$  the list of the original property instances that are successfully covered by a set of patterns  $\Pi$ :

$$Instance(p, \Pi) = \bigcup_{pa \in \Pi} (Cov_p(p, pa) \cap instances(p)) \quad (23)$$

The  $instances(p)$  denotes the list of original instances which have the property  $p$  in original KB  $D$ . Thus, the instance property recall  $InstanceRec(p, \Pi)$  defined as:

$$InstanceRec(p, \Pi) = \frac{|instances(p, \Pi) \cap instances(p)|}{|instances(p)|} \quad (24)$$

The overall recall at the instance property level  $InstanceRec_{PropertyAll}$  is computed as the weighted mean of the various instance property recall  $InstanceRec$  for all the properties of the ground-truth.

$$InstanceRec_{PropertyAll} = \frac{\sum_{p \in P} wi(p) * InstanceRec(p, \Pi)}{|P|} \quad (25)$$

The  $wi(p)$  is the weight of the property  $p$  and it measures the percentage of instances of a property  $p$  with respect to the total number of property instances in the KB. It is defined as the number of instances of property  $p$  in the KB  $instances(p)$  compared to the total number of property instances in the KB. Again the idea here is

to capture the important properties by weighting in the number of property instances it represents.

$$wi(p) = \frac{instances(p)}{\sum_{p1 \in P} instances(p1)} \quad (26)$$

We define  $InstancePrec(p, \Pi)$ , the precision of a property  $p$  in  $P$  over the set of patterns  $\Pi$  as follows:

$$InstancePrec(p, \Pi) = \frac{|instances(p) \cap instances(p, \Pi)|}{|instances(p, \Pi)|} \quad (27)$$

Thus, we define the overall instance precision for property instances denoted by  $InstancePrec_{PropertyAll}$  as the mean of the various  $InstancePrec(c, \Pi)$  for all the properties of the ground-truth Schema  $S$ :

$$InstancePrec_{PropertyAll} = \frac{\sum_{p \in P} InstancePrec(p, \Pi)}{|P1|} \quad (28)$$

where  $P1 \in P$  is the list of retrieved properties, or in other words the list of properties having  $InstancePrec(p, \Pi) > 0$ . The overall instance recall and the overall instance precision for property instances are combined by the instance class F-Measure, namely  $SchemaF1_c$ :

$$InstanceF1_p = 2 * \frac{InstancePrec_{PropertyAll} * InstanceRec_{PropertyAll}}{InstancePrec_{PropertyAll} + InstanceRec_{PropertyAll}} \quad (29)$$

Thus, the overall instance F-measure  $InstanceF1$  is obtained by combining the overall instance schema F-Measure  $InstanceF1_c$  and overall property instance F-Measure  $InstanceF1_p$ .

$$InstanceF1 = \beta * InstanceF1_p + (1 - \beta) * InstanceF1_c \quad (30)$$

where the weight  $\beta \in [0, 1]$ . The overall instance F-measure can be viewed as a compromise between overall class instance F-Measure and overall property instance F-Measure. It is high only when both overall class and property instance F-Measure are high. It is equivalent to the class instance F-Measure when  $\beta = 0$  and to the property instance F-Measure when  $\beta = 1$ .

**Connectivity** One more important aspect that we need to consider, is the connectivity of the summary, i.e. is the summary a connected graph? So, we propose a new metric to measure how many disconnected graphs exist in the summary and what percentage of

the classes in the ground truth they represent. The connectivity of a summary graph  $G_s$ ,  $Con(G_s)$  is defined as the number of the connected components (independent subgraphs) of the summary graph divided on the number of the connected components (independent subgraphs) of the ground truth.

$$Con(G_s) = \frac{\text{number of connected components of the summary}}{\text{number of connected components of the ground truth}} \quad (31)$$

This metric gives an indication of the connectivity of a generated summary. If it is 1, it shows that the summary is a fully connected graph comparing to the ground truth, while when it is bigger than 1 but not so high it means that the summary graph miss some links between the original classes of the ground truth. But if it is so high this means that the summary miss all or almost all the links between the original classes which means we ended-up with a disconnected summary graph. Thus, this metric provides a better understanding of how the interlinked classes of the original KB are represented in the summary, which might be of help in cases where a user has two summaries with the similar or almost similar values for the rest of the previously discussed measures.

## 5. Representative Algorithms for validating the Quality Framework

As we have already mentioned in section 3 the RDF graph summarization could be grouped into four main categories. Based on the results reported in the literature we have chosen three of the most well performing RDF graph summarization algorithms [17,12,33] according to their authors. Our selection of these algorithms was also based on specific properties and features that they demonstrate: (a) they do not require the presence of RDF schema (triples) in order to work properly, (b) they work on both homo- and heterogeneous KBs, (c) they provide statistical information about the available data (which can be used to estimate a query's expected results' size), and (d) they provide a summary graph that is considerably smaller than the original graph.

**ExpLOD** [17,18] is a RDF graph summarization algorithm and tool that produces summary graphs for specific aspects of an RDF dataset, like class or predicate usage. The summary graph is computed over the RDF graph based on a forward bisimulation that cre-

ates group nodes based on classes and predicates. Two nodes  $v$  and  $u$  are bisimilar if they have the same set of types and properties. The generated summaries contain metadata about the structure of the RDF graph, like the sets of used RDF classes and properties. Some statistics like the number of instances per class or per property are aggregated with this structural information. The ExpLOD summaries are extracted by partition refinement algorithms or alternatively via SPARQL query where the summary graph is a labeled graph with unlabeled edges. The advantage of ExpLOD approach is that its generated summaries show a dataset's structure as homo- or heterogeneous as it may be. The big disadvantage is the need for transforming the original RDF KB into a ExpLOD graph which is an unlabeled edges graph, where for each triple in RDF KB it generates a node for the subject, node for the object and a unique node for the predicate. Then an edge is drawn from the subject node to the predicate node and other edge from the predicate node to the object node. This process requires the materialization of the whole dataset and this can be limiting in cases of large KBs. The second limitation is that the created summary is not necessarily a RDF graph itself.

**Campinas et al** [12] are creating their own RDF summarization graph, whose nodes represent a subset of the original nodes based on their types or used predicates. This summary graph is generated by the following mechanism: (1) extract the types and predicates for each node in the original graph; (2) group the nodes which share the same set of types into the same node summary where two nodes, one of type A and one of types A and B, will end up in different disjoint summary nodes; (3) group based on attributes only if a node does not have a class definition. Like ExpLOD, a summary node is created for each combination of classes, i.e., two nodes, one of type A and one of types A and B, will end up in different disjoint summary nodes. Some statistics like the number of instances per class or the number of property instances are aggregated with this summary graph. Unlike ExpLOD, the summary nodes are not further partitioned based on their interlinks (properties), i.e., two nodes of type A, one has a, b and c properties and one has a and d properties will end up in the same summary node. Unlike ExpLOD, their summary graph is a RDF graph which makes it compatible for storing at RDF databases and queried by SPARQL.

**Zneika et al.** [33,34] present an approach for RDF graph summarization based on mining a set of approximate graph patterns is presented. It aims at extract-

ing the best approximate RDF graph patterns that describe the input dataset and it works in three independent steps that are described below.

*Binary Matrix Mapper:* Transform the RDF graph into a binary matrix, where the rows represent the subjects and the columns represent the predicates. They preserve the semantics of the information by capturing distinct types (if present), all attributes and properties (capturing property participation both as subject and object for an instance).

*Graph Pattern Identification:* The binary matrix created in previous step is used in a calibrated version of the PaNda+ [24] algorithm, which allows to experiment with different cost functions while retrieving the best approximate RDF graph patterns. Each extracted pattern identifies a set of subjects (rows) all having approximately the same properties (cols). The patterns are extracted so as to minimize errors and to maximize the coverage (i.e. provide a richer description) of the input data. A pattern thus encompasses a set of concepts (type, property, attribute) of the RDF dataset, holding at the same time information about the number of instances that support this set of concepts.

*Constructing the RDF summary graph:* A process, which reconstructs the summary as a valid RDF graph using the extracted patterns is applied at the end. The process exploits information already embedded in the binary matrix and constructs a valid RDF schema to summarize the KB.

## 6. Experiments

In this section, we compare the quality of the generated summaries of the three RDF graph summarization approaches covered in section 5. We implemented these three approaches in Java 1.8 using the Nxparser<sup>2</sup> API to parse the RDF triples. All the experiments ran on a Intel(R) Core(i5) Opteron 2.5 GHz server with 16 GB of RAM (of which 14 GB was assigned to the Java Virtual Machine), running Windows 7. Section 6.1 describes the datasets considered in the experiments. Section 6.2 gives a quality evaluation of the created summaries based on the three discussed approaches and using the metrics described in section 4.

### 6.1. Datasets

Table 3 shows the datasets from the LOD cloud that are considered for the experiments. The first seven columns show the following information about each dataset: its name, the number of triples it contains, and the number of instances, classes, predicates, properties and attributes. The eighth column shows the class instance distribution metric which provides an indication on how instances are spread across the classes and it is defined as the standard deviation (SD) in the number of instances per class. When the number of class instances per class in a dataset is quite close then the standard deviation is small; while, when there are considerable differences, the standard deviation will be relatively large. The ninth column shows the property instance distribution metric which provides an indication on how instances are spread across the properties and it is also defined as standard deviation (SD) in the number of instances per property.

The main goal of our datasets selection is to use real-world datasets from diverse domains with different size (number of triples) and with different numbers of classes (and class instances) and properties (and properties instances). We are also interested in the distribution of the data which might indicate if the structure of the KB or the size of the represented knowledge could affect the quality of the generated summaries. So we have datasets from 270 thousand (Jpeel) to 263 million triples (Lobid), from one (Bank2) to 53 unique classes (LinkedMDB), from about 76 thousand (Jpeel) to about 18 million unique instances/entities and from 12 to 222 predicates. These datasets range from being very homogeneous (the Bank dataset where all subjects have the same list of attributes and properties) to being very heterogeneous (LinkedMDB where the attributes and properties are very heterogeneous across types). The diversity of the datasets can help us to understand better how the selected approaches work in different situations and thus validate that the proposed quality metrics will capture the different behaviours correctly.

### 6.2. Evaluation Results

In this section, we discuss the quality results of the RDF graph summarization approaches covered in section 5, evaluated over all the datasets described in table 3 for the following two cases:

- Typed Dataset: A significant number of instances of a dataset have at least one type of link/property.

<sup>2</sup>Nxparser: <https://github.com/nxparser/nxparser>

Dataset	Triples	Instances	Classes	Predicates	properties	attributes	Class instance distribution		Property instance distribution	
							Mean	SD	Mean	SD
Jpeel [4]	271,369	76,229	9	26	14	12	8,449	8,289.61	9,374.48	15,988.21
Jamendo [3]	1,047,950	335,925	11	25	14	11	20,542	19,622.08	34,633.48	59,458.62
Sec <sup>a</sup>	1,813,135	460,446	5	12	3	9	66,861.8	41,233.64	144,041.83	63,388.13
linkedMDB [6]	6,148,121	694,400	53	222	153	69	13,971	37,368.26	24,758.70	80,271.76
Bank [1]	7,348,860	200,429	1	33	0	33	200,429	0	197,065.61	4,786.98
Wordnet [8]	8,574,807	647,215	5	63	55	8	129,147	69,768.22	59,947.92	113,775.88
DBLP [2]	41,802,523	5,942,858	10	19	9	10	497,153.9	971,029.76	538,837.42	805,531.71
Linkedet [5]	49,084,152	5,364,776	30	121	44	77	178,826	217,293.64	214,010.65	218,145.29
Lobid [7]	263,215,517	17,854,885	24	104	40	64	663,355.26	996,359.95	661,974.82	979,956.84

Table 3: Descriptive statistics of the datasets

<sup>a</sup>U.S. SEC data: <http://www.govtrack.us/data/misc/sec.n3.gz>

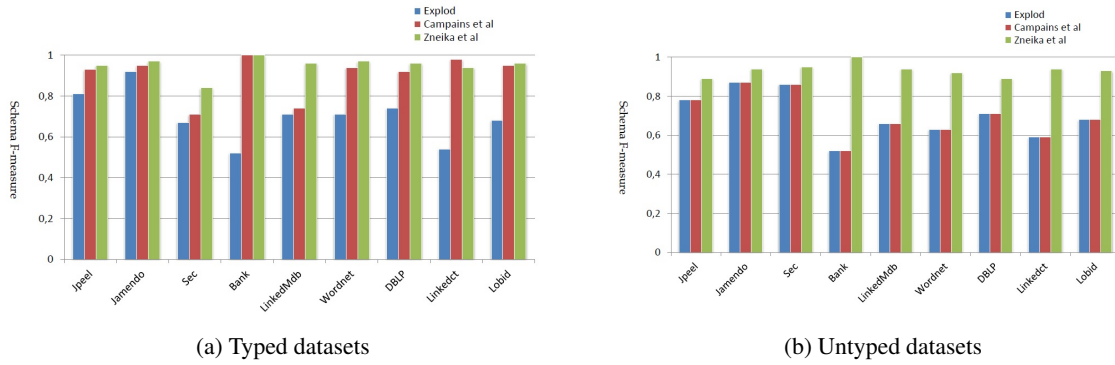


Fig. 2.: F-Measure results for typed/untyped presented datasets at the schema Level

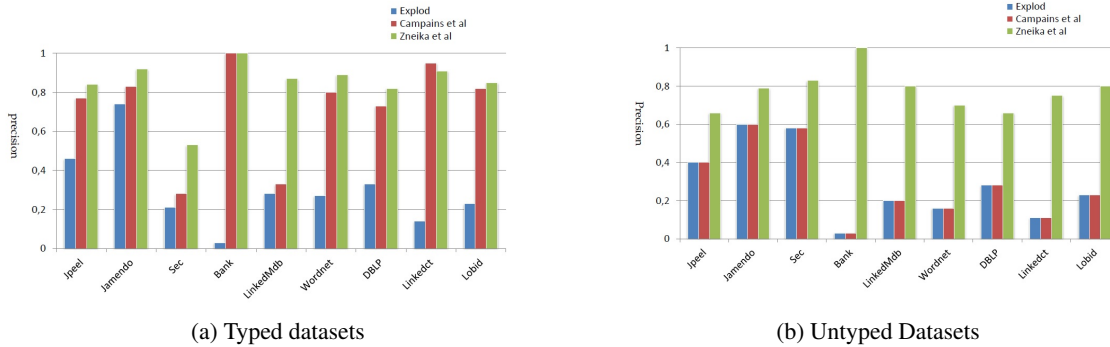


Fig. 3.: Class precision results for typed/untyped presented datasets at the schema Level

- Untyped Dataset: None of the datasets subjects/objects or properties has a defined type (we explicitly deleted all of them).

#### 6.2.1. Schema level

Table 4 reports the precision, recall and F-Measure values at the schema level for classes and properties of

the generated RDF summaries over the set of datasets depicted in table 3 for the *typed* and *untyped* cases. The left part of Table 4 shows the results for the *typed* used datasets while the right part shows the results for *untyped* used datasets. The Figures 2 and 3 are a flow chart representing for The overall schema F-Measure

Algorithm	$R_c$	$P_c$	$F1_c$	$R_p$	$P_p$	$F_p$	F1
ExpLod	1	0.46	0.63	1	1	1	0.81
Campinas et al	1	0.77	0.87	1	1	1	0.93
Zneika et al	1	0.84	0.90	1	1	1	0.95

(a) Typed Jpeel

Algorithm	$R_c$	$P_c$	$F1_c$	$R_p$	$P_p$	$F_p$	F1
ExpLod	1	0.74	0.85	1	1	1	0.92
Campinas et al	1	0.83	0.90	1	1	1	0.95
Zneika et al	1	0.92	0.95	1	1	1	0.97

(c) Typed Jamendo

Algorithm	$R_c$	$P_c$	$F1_c$	$R_p$	$P_p$	$F_p$	F1
ExpLod	1	0.21	0.34	1	1	1	0.67
Campinas et al	1	0.28	0.43	1	1	1	0.71
Zneika et al	1	0.53	0.69	1	1	1	0.84

(e) Typed Sec

Algorithm	$R_c$	$P_c$	$F1_c$	$R_p$	$P_p$	$F_p$	F1
ExpLod	1	0.03	0.05	1	1	1	0.52
Campinas et al	1	1	1	1	1	1	1
Zneika et al	1	1	1	1	1	1	1

(g) Typed Bank

Algorithm	$R_c$	$P_c$	$F1_c$	$R_p$	$P_p$	$F_p$	F1
ExpLod	1	0.28	0.43	1	1	1	0.71
Campinas et al	1	0.33	0.49	1	1	1	0.74
Zneika et al	1	0.87	0.93	1	1	1	0.96

(i) Typed LinkedMDB

Algorithm	$R_c$	$P_c$	$F1_c$	$R_p$	$P_p$	$F_p$	F1
ExpLod	1	0.27	0.42	1	1	1	0.71
Campinas et al	1	0.80	0.88	1	1	1	0.94
Zneika et al	1	0.89	0.94	1	1	1	0.97

(k) Typed Wordnet

Algorithm	$R_c$	$P_c$	$F1_c$	$R_p$	$P_p$	$F_p$	F1
ExpLod	1	0.33	0.49	1	1	1	0.74
Campinas et al	1	0.73	0.84	1	1	1	0.92
Zneika et al	1	0.82	0.90	1	1	1	0.96

(m) Typed DBLP

Algorithm	$R_c$	$P_c$	$F1_c$	$R_p$	$P_p$	$F_p$	F1
ExpLod	1	0.14	0.09	1	1	1	0.54
Campinas et al	1	0.95	0.97	1	1	1	0.98
Zneika et al	0.95	0.91	0.92	0.93	1	0.96	0.94

(o) Typed Linkedct

Algorithm	$R_c$	$P_c$	$F1_c$	$R_p$	$P_p$	$F_p$	F1
ExpLod	1	0.23	0.37	1	1	1	0.68
Campinas et al	1	0.82	0.90	1	1	1	0.95
Zneika et al	1	0.85	0.91	1	1	1	0.96

(q) Typed Lobid

Algorithm	$R_c$	$P_c$	$F1_c$	$R_p$	$P_p$	$F_p$	F1
ExpLod	1	0.40	0.57	1	1	1	0.78
Campinas et al	1	0.40	0.57	1	1	1	0.78
Zneika et al	1	0.66	0.79	1	1	1	0.89

(b) Untyped Jpeel

Algorithm	$R_c$	$P_c$	$F1_c$	$R_p$	$P_p$	$F_p$	F1
ExpLod	1	0.60	0.75	1	1	1	0.87
Campinas et al	1	0.60	0.75	1	1	1	0.87
Zneika et al	1	0.79	0.88	1	1	1	0.94

(d) Untyped Jamendo

Algorithm	$R_c$	$P_c$	$F1_c$	$R_p$	$P_p$	$F_p$	F1
ExpLod	1	0.58	0.73	1	1	1	0.86
Campinas et al	1	0.58	0.73	1	1	1	0.86
Zneika et al	1	0.83	0.90	1	1	1	0.95

(f) Untyped Sec

Algorithm	$R_c$	$P_c$	$F1_c$	$R_p$	$P_p$	$F_p$	F1
ExpLod	1	0.03	0.05	1	1	1	0.52
Campinas et al	1	0.03	0.05	1	1	1	0.52
Zneika et al	1	1	1	1	1	1	1

(h) Untype Bank

Algorithm	$R_c$	$P_c$	$F1_c$	$R_p$	$P_p$	$F_p$	F1
ExpLod	1	0.20	0.33	1	1	1	0.66
Campinas et al	1	0.20	0.33	1	1	1	0.66
Zneika et al	1	0.80	0.89	1	1	1	0.94

(j) Untyped LinkedMDB

Algorithm	$R_c$	$P_c$	$F1_c$	$R_p$	$P_p$	$F_p$	F1
ExpLod	1	0.16	0.27	1	1	1	0.63
Campinas et al	1	0.16	0.27	1	1	1	0.63
Zneika et al	1	0.70	0.85	1	1	1	0.92

(l) Untyped Wordnet

Algorithm	$R_c$	$P_c$	$F1_c$	$R_p$	$P_p$	$F_p$	F1
ExpLod	1	0.28	0.43	1	1	1	0.71
Campinas et al	1	0.28	0.43	1	1	1	0.71
Zneika et al	1	0.66	0.79	1	1	1	0.89

(n) Untyped DBLP

Algorithm	$R_c$	$P_c$	$F1_c$	$R_p$	$P_p$	$F_p$	F1
ExpLod	1	0.11	0.19	1	1	1	0.59
Campinas et al	1	0.11	0.19	1	1	1	0.59
Zneika et al	1	0.75	0.85	1	1	1	0.94

(p) Untyped Linkedct

Algorithm	$R_c$	$P_c$	$F1_c$	$R_p$	$P_p$	$F_p$	F1
ExpLod	1	0.23	0.37	1	1	1	0.68
Campinas et al	1	0.23	0.37	1	1	1	0.68
Zneika et al	1	0.80	0.87	1	1	1	0.93

(r) Untyped Lobid

Table 4: Precision, Recall and F-Measure at the Schema level. The  $R_c$  column reports the schema class Recall  $SchemaRec_{ClassAll}$ . The  $P_c$  column reports the schema class precision  $SchemaPrec_{ClassAll}$ . The  $F1_c$  reports the schema class F-measure  $SchemaF1_c$ . The  $R_p$  column reports the schema property Recall  $SchemaRec_{PropertyAll}$ . The  $P_p$  column reports the schema property precision  $SchemaPrec_{PropertyAll}$ . The  $F1_p$  column reports the schema property F-measure  $SchemaF1_p$ . The F1 column reports the overall schema F-Measure  $SchemaF1$ .

and the class precision metrics values receptively that carries more visualization details.

We can note from Table 4 that the schema property recall, schema property precision and the schema property F-Measure, reported in columns  $R_p$ ,  $P_p$  and  $F_p$  respectively, are always equal to 1 for the ExpLod and the Campinas et al algorithms over all the presented datasets. The same is true for the schema class recall reported in column  $R_c$ . We can also note from the right part of the 4 that the values of the previously mentioned measures are equal to 1. This is because the ExpLod and Campinas et al algorithms depend on the notion of the forward bisimulation that groups the original nodes based on classes and/or predicates, hence they are no missed properties or types (and of course nothing new is added), thus the schema class recall values will be always 1 for the ExpLod and the Campinas et al. A predicates-based grouping is necessary for the Campinas et al algorithm when the entities' nodes do not have a class definition, hence they are no missed properties for the *untyped* case, which explains why the values for these measures have not changed for the *untyped* datasets. This is also explains why we have the same measures' values for the ExpLod and Campinas et al for the *untyped* datasets. For Zneika et al algorithm, although it depends on the approximation type selected, if we exclude the linkedct dataset the values for measures mentioned previously are also equal to 1 for the *typed* an *untyped* datasets, which means that the algorithm successfully summarizes the KBs, despite the fact that by construction the algorithm uses approximate pattern mining to detect the classes and properties available and thus some could have been possibly missed.

Another notable observation from the Table 4g and the Figure 2b, is that for the Bank dataset and for the overall schema F-Measure the perfect value (equal to 1) is reported for the Zneika et al and Campinas et al algorithms. This is because the Bank dataset is a fully *typed* and homogeneous dataset(each subject of this dataset has at least one typeof link/property) and as we explained earlier, the Campinas et al algorithm groups the original nodes based only on their types when types exist, hence they are no missed or added properties in this case.

For the Sec dataset, the table 4e shows that the values of schema class precision reported in column  $P_c$  and depicted in Figure 3a are low for the three discussed algorithms. This is because that the ground truth schema of the Sec dataset contains a lot of inheritance relationships and as none of three discussed al-

gorithms deals with inheritance, the three algorithms end up with a lot of overlapping patterns (some properties which belong to the subclasses are assigned to the patterns which represent the superclasses).

Table 4 shows well that algorithms like ExpLod do not provide quality summaries in extreme cases like the Bank dataset (where we have only one class) or in heterogeneous datasets like LinkedMDB, Linkedct and DBLP, where we report very low precision values, because instances of the same class in these cases are having quite different properties and they cannot be grouped together by ExpLod. We can also note from Table 4 and the Figures 3 and 2 that the Zneika et al algorithm gives better results, when compared with the other two algorithms, over all the presented datasets, and it showcases that it works well with heterogeneous datasets like the LinkedMdb, unlike the ExpLod and Campinas et al that give a low class precision with the heterogeneous datasets.

By comparing the results for the *typed* datasets case depicted in Figure 3a and the *untyped* datasets depicted in Figure 3b. We can easily observe that the behaviour of Zneika et al and ExpLod algorithms in the case of the *untyped* cases is the same as in the case of the *typed* datasets, which means that the quality of the summary is not affected by the presence (or not) of schema information in the KB. While we can easily observe the significant impact the absence of typeof schema information had for the Campinas et al algorithm.

The discussion so far provides some insights on how we can use the proposed Quality Framework to assess the quality of the summaries produced by the different algorithms. Since we are looking at comparing the quality of the computed summary to a ground truth summary provided by an expert in general we can observe that:

- the summarization algorithms usually capture correctly the properties involved in the data but miss at different levels (and for different reasons) some of the classes. The Quality Framework provides enough resolution to really identify the algorithms that provide a better summary in turn of the classes reported and the quality of this report (e.g. are all properties reported, is the class present as one entity in the computed summary, etc.).
- the summarization algorithms could have quite a few differences when reporting on the contents of the KB and the quality of the summaries could greatly vary and this is mostly because of the dif-

ferences in the precision of reporting the classes in the summary, including penalizing verbose descriptions (like those reported by Explod). So actually we can capture even fine differences where for example a single class in the ground truth is represented by two in the computed summary.

#### 6.2.2. Instance level

Table 5 reports the precision, the recall and the F-Measure of RDF summaries at the instance level, based on the same datasets and algorithms as before. The left part of Table 4 shows the results for the *typed* datasets while the right part shows the results for *untyped* datasets. For each dataset, we report the precision, the recall and the F-measure values at class and property level. We note that Explod produces the best results (actually perfect ones, always 1) since it is not missing any property or class instance because Explod works by grouping of even two instances if they have the same set of attributes and types, thus does not add any false positives. We can also note that the instance class precision and the instance recall precision reported in columns  $P_c$  and  $R_c$  are always equal to 1 for Campinas et al algorithm over all the presented datasets, while the property instance precision reported in column  $P_p$  is low in most presented datasets. This is because the Campinas et al algorithm works by grouping of two instances if they have the same set types, thus it does not add any false positives at the class level but maybe it will assign some properties to subjects/instances which do not actually have these properties at the KB (false positive at the property level). This is explain why it is important to take into consideration quality metrics at the property and class level.

Table 5 shows also that the behavior of Zneika et al and Explod algorithms in the case of the *untyped* datasets is the same or approximately the same as in the case of the *typed* datasets, which means that the quality of the summary with regard to the coverage of the instances is not affected by the presence (or not) of schema information in the KB for these two algorithms. On the other hand, we can easily observe the great positive impact left by the absence of typeof schema information for the Campinas et al algorithm.

From this discussion, we can observe that the summarization algorithms provide results of good quality when the coverage of the instances in the KB is concerned. The proposed quality metrics clearly show that relying only on this metric is not adequate to judge the quality of a summary since a lot of the algorithms report perfect scores in all measures. But still we have

cases where we can distinguish the quality among the results based on the instances covered by the computed summary, especially when algorithms use approximative methods to compute the summary (one algorithm in our case). It is worth noting here that our Quality Framework can capture both under-coverage (when not all instances are represented in the final result) and over-coverage (when some instances are represented more than once or some fictitious instances are included) of instances. With the metrics at the instance level we can capture these fine differences for covering correctly or not and how much the instance in the KB.

#### 6.2.3. Combined results

By comparing the results in both cases, it becomes clear why it is important, to take into consideration quality metrics that capture information both at the instance and the conceptual level. Otherwise behaviors like the one demonstrated by Explod cannot be captured and summaries that are flawed might be indistinguishable from better ones. Overall, we could argue that the Quality Framework introduced in section 4 is adequate for capturing the fine differences in quality of the summaries produced by the three algorithms. We can also see that with a closer look at the results we can gain or verify insights on how specific algorithms work and the quality of the summaries they produce.

One final metric to be considered is whether the final graph is connected or not and appears as more than one connected components. This might mean that the summarization algorithm while captures correctly the important propeties and classes in the KB fails to provide at the end a connected graph. This is important because this might signify whether the summary graph is usable or not for answering for example SPARQL queries. Table 6 reports the connectivity metric values for the summaries produced by the three discussed algorithms over all the datasets described in table 3. It shows that the Explod has always a high values for this metric which means it provides a disconnected summary while the two others always have always 1 which means that these two algorithm provide a fully connected summary.

So measuring the quality at the schema level, the instance level and the connected components of the graph can give us a detailed view of the strengths and weaknesses of a summary and decide whether to use it or not depending on the potential use and application. We avoided combining all the measures together because this might blur the final picture. The idea is

Algorithm	$R_c$	$P_c$	$F1_c$	$R_p$	$P_p$	$F1_p$	F1
ExpLod	1	1	1	1	1	1	1
Campinas et al	1	1	1	1	0.33	0.49	0.74
Zneika et al	0.99	0.96	0.97	0.99	0.95	0.97	0.97

(a) Typed Jpeel

Algorithm	$R_c$	$P_c$	$F1_c$	$R_p$	$P_p$	$F1_p$	F1
ExpLod	1	1	1	1	1	1	1
Campinas et al	1	1	1	1	0.49	0.65	0.82
Zneika et al	1	0.98	0.99	1	0.98	0.99	0.99

(b) Untyped Jpeel

Algorithm	$R_c$	$P_c$	$F1_c$	$R_p$	$P_p$	$F1_p$	F1
ExpLod	1	1	1	1	1	1	1
Campinas et al	1	1	1	1	1	1	1
Zneika et al	1	0.98	0.99	1	0.98	0.99	0.99

(c) Typed Jamendo

Algorithm	$R_c$	$P_c$	$F1_c$	$R_p$	$P_p$	$F1_p$	F1
ExpLod	1	1	1	1	1	1	1
Campinas et al	1	1	1	1	0.92	0.95	0.97
Zneika et al	1	1	1	1	1	1	1

(d) Untyped Jamendo

Algorithm	$R_c$	$P_c$	$F1_c$	$R_p$	$P_p$	$F1_p$	F1
ExpLod	1	1	1	1	1	1	1
Campinas et al	1	1	1	1	1	1	1
Zneika et al	1	1	1	1	1	1	1

(e) Typed Sec

Algorithm	$R_c$	$P_c$	$F1_c$	$R_p$	$P_p$	$F1_p$	F1
ExpLod	1	1	1	1	1	1	1
Campinas et al	1	1	1	1	0.97	0.98	0.99
Zneika et al	1	1	1	1	0.97	0.98	0.99

(f) Untyped Sec

Algorithm	$R_c$	$P_c$	$F1_c$	$R_p$	$P_p$	$F1_p$	F1
ExpLod	1	1	1	1	1	1	1
Campinas et al	1	1	1	1	1	1	1
Zneika et al	1	1	1	1	0.97	0.98	0.99

(g) Typed Bank

Algorithm	$R_c$	$P_c$	$F1_c$	$R_p$	$P_p$	$F1_p$	F1
ExpLod	1	1	1	1	1	1	1
Campinas et al	1	1	1	1	0.08	0.14	0.57
Zneika et al	1	0.93	0.96	1	0.73	0.84	0.89

(h) Untyped Bank

Algorithm	$R_c$	$P_c$	$F1_c$	$R_p$	$P_p$	$F1_p$	F1
ExpLod	1	1	1	1	1	1	1
Campinas et al	1	1	1	1	1	1	1
Zneika et al	1	0.93	0.96	1	0.73	0.84	0.89

(i) Typed LinkedMDB

Algorithm	$R_c$	$P_c$	$F1_c$	$R_p$	$P_p$	$F1_p$	F1
ExpLod	1	1	1	1	1	1	1
Campinas et al	1	1	1	1	0.32	0.48	0.74
Zneika et al	1	0.80	0.88	1	0.82	0.90	0.89

(j) Untyped LinkedMDB

Algorithm	$R_c$	$P_c$	$F1_c$	$R_p$	$P_p$	$F1_p$	F1
ExpLod	1	1	1	1	1	1	1
Campinas et al	1	1	1	1	1	1	1
Zneika et al	1	0.93	0.96	1	0.73	0.84	0.89

(k) Typed Wordnet

Algorithm	$R_c$	$P_c$	$F1_c$	$R_p$	$P_p$	$F1_p$	F1
ExpLod	1	1	1	1	1	1	1
Campinas et al	1	1	1	1	0.64	0.78	0.89
Zneika et al	1	0.82	0.90	1	0.71	0.83	0.86

(l) Untyped Wordnet

Algorithm	$R_c$	$P_c$	$F1_c$	$R_p$	$P_p$	$F1_p$	F1
ExpLod	1	1	1	1	1	1	1
Campinas et al	1	1	1	1	0.79	0.88	0.94
Zneika et al	1	1	1	1	0.96	0.98	0.99

(m) Typed DBLP

Algorithm	$R_c$	$P_c$	$F1_c$	$R_p$	$P_p$	$F1_p$	F1
ExpLod	1	1	1	1	1	1	1
Campinas et al	1	1	1	1	0.79	0.88	0.94
Zneika et al	1	1	1	1	0.96	0.98	0.99

(n) Untyped DBLP

Algorithm	$R_c$	$P_c$	$F1_c$	$R_p$	$P_p$	$F1_p$	F1
ExpLod	1	1	1	1	1	1	1
Campinas et al	1	1	1	1	1	1	1
Zneika et al	1	0.93	0.96	1	0.73	0.84	0.89

(o) Typed Linkedct

Algorithm	$R_c$	$P_c$	$F1_c$	$R_p$	$P_p$	$F1_p$	F1
ExpLod	1	1	1	1	1	1	1
Campinas et al	1	1	1	1	0.37	0.54	0.77
Zneika et al	1	0.91	0.95	1	0.86	0.92	0.935

(p) Untyped Linkedct

Algorithm	$R_c$	$P_c$	$F1_c$	$R_p$	$P_p$	$F1_p$	F1
ExpLod	1	1	1	1	1	1	1
Campinas et al	1	1	1	1	1	1	1
Zneika et al	1	0.89	0.94	1	0.77	0.88	0.91

(q) Typed Lobid

(r) Untyped Lobid

Table 5: Precision, Recall and F-Measure at the instance level. The  $R_c$  column reports the instance class Recall  $InstanceRec_{ClassAll}$ . The  $P_c$  column reports the instance class precision  $InstancePrec_{ClassAll}$ . The  $F1_c$  column reports the instance class F-measure  $InstanceF1_c$ . The  $R_p$  column reports the instance property Recall  $InstanceRec_{PropertyAll}$ . The  $P_p$  column reports the instance property precision  $InstancePrec_{PropertyAll}$ . The  $F1_p$  column reports the instance property F-measure  $InstanceF1_p$ . The F1 column reports the overall instance F-Measure  $InstanceF1$ .



Dataset	ExpLod	Campinas et al	Zneika et al
Jpeel	25	1	1
Jamendo	31	1	1
Sec	6	1	1
LinkedMDB	8464	1	1
Bank	11	1	1
Wordnet	778	1	1
DBLP	108	1	1
Linkedct	5699	1	1
Lobid	9786	1	1

Table 6: Connectivity Metric results

not to necessarily prove an algorithm as better or worse (we can do this to a great extent through the different *F-measures*) but mainly to help the user understand the different qualities of the summaries and choose the best one for the different needs of the diverse use cases.

## 7. Conclusions and Future Work

In this paper, we provided a quality framework by introducing a set of metrics, that can be used to comprehensively evaluate any RDF summarization algorithm that is reported in the literature. The metrics proposed are independent of the algorithm, the KB (thus the data) and the existence or not of schema information within the KB. The literature does not report any other effort that tries to capture the quality properties of RDF graph summaries both at the concept (schema) and instance level in a complete and comprehensive way. The experiments showed that using the proposed set of metrics we are able now to compare the quality at different levels of the RDF summaries produced by different algorithms found in the literature, applied on different and diverse datasets and extract useful insights for their suitability for various tasks.

We plan to extend this work by applying the framework to Linked Data sources where quality results might be different for each part of the linked datasets. We would like to explore both theoretically and experimentally whether there are ways to provide consolidated quality metrics treating the linked KBs as one, which will go beyond simply averaging the individual quality results. We would also like to use the framework to assess the quality of the results of more algorithms, in order to validate experimentally its suitability.

## References

- [1] About World Bank Linked Data: World Bank Finances. <http://worldbank.270a.info/about.html#about-datasets/>. Accessed: 2017-03-30.
- [2] DBLP Bibliography Database in RDF Datahub. <https://datahub.io/dataset/fu-berlin-dblp>. Accessed: 2017-03-30.
- [3] Jamendo DBTune home,. <http://dbtune.org/jamendo>. Accessed: 2017-03-30.
- [4] John Peel DBTune home,. <http://dbtune.org/bbc/peel>. Accessed: 2017-03-30.
- [5] LinkedCT Datahub. <https://datahub.io/dataset/linkedct/>. Accessed: 2017-03-30.
- [6] LinkedMDB home,. <http://www.linkedmdb.org/>. Accessed: 2017-03-30.
- [7] lobid-Bibliographic Resources. <https://datahub.io/dataset/lobid-resources>. Accessed: 2017-03-30.
- [8] WordNet RDF home. <http://wordnet-rdf.princeton.edu>. Accessed: 2017-03-30.
- [9] Anas Alzogbi and Georg Lausen. Similar structures inside rdf graphs. In *LDOW*, 2013.
- [10] Samur Araujo, Jan Hidders, Arjen P de Vries, and Daniel Schwabe. Serimi: resource description similarity, rdf instance matching and interlinking. In *Proceedings of the 6th International Conference on Ontology Matching-Volume 814*, pages 246–247. CEUR-WS. org, 2011.
- [11] Stéphane Campinas, Renaud Delbru, and Giovanni Tummarello. Efficiency and precision trade-offs in graph summary algorithms. In *Proceedings of the 17th International Database Engineering & Applications Symposium*, pages 38–47. ACM, 2013.
- [12] Stephane Campinas, Thomas E Perry, Diego Ceccarelli, Renaud Delbru, and Giovanni Tummarello. Introducing rdf graph summary with application to assisted sparql formulation. In *Database and Expert Systems Applications (DEXA), 2012 23rd International Workshop on*, pages 261–266. IEEE, 2012.
- [13] Mariano P Consens, Valeria Fionda, Shahana Khatchadourian, and Giuseppe Pirro. S+ epps: construct and explore bisimulation summaries, plus optimize navigational queries; all on existing sparql systems. *Proceedings of the VLDB Endowment*, 8(12):2028–2031, 2015.
- [14] François Goasdoué and Ioana Manolescu. Query-oriented summarization of rdf graphs. *Proceedings of the VLDB Endowment*, 8(12), 2015.
- [15] Amit Krishna Joshi, Pascal Hitzler, and Guozhu Dong. Towards logical linked data compression. In *Proceedings*

- of the Joint Workshop on Large and Heterogeneous Data and Quantitative Formalization in the Semantic Web, LHD+SemQuant2012, at the 11th International Semantic Web Conference, ISWC2012. Citeseer, 2012.
- [16] Amit Krishna Joshi, Pascal Hitzler, and Guozhu Dong. Logical linked data compression. In *Extended Semantic Web Conference*, pages 170–184. Springer, 2013.
  - [17] Shahan Khatchadourian and Mariano P Consens. Explod: Summary-based exploration of interlinking and rdf usage in the linked open data cloud. *The Semantic Web: Research and Applications*, pages 272–287, 2010.
  - [18] Shahan Khatchadourian and Mariano P Consens. Exploring rdf usage and interlinking in the linked open data cloud using explod. In *LDOW*, 2010.
  - [19] Shahan Khatchadourian and Mariano P Consens. Understanding billions of triples with usage summaries. *Semantic Web Challenge*, 2011.
  - [20] Shahan Khatchadourian and Mariano P Consens. Constructing bisimulation summaries on a multi-core graph processing framework. In *Proceedings of the GRADES'15*, page 8. ACM, 2015.
  - [21] Mathias Konrath, Thomas Gottron, and Ansgar Scherp. Schemex—web-scale indexed schema extraction of linked open data. *Semantic Web Challenge, Submission to the Billion Triple Track*, pages 52–58, 2011.
  - [22] Mathias Konrath, Thomas Gottron, Steffen Staab, and Ansgar Scherp. Schemex—efficient construction of a data catalogue by stream-based indexing of linked data. *Web Semantics: Science, Services and Agents on the World Wide Web*, 16:52–58, 2012.
  - [23] Amine Louati, Marie-Aude Aufaure, Yves Lechevallier, and France Chatenay-Malabry. Graph aggregation: Application to social networks. In *HSDA*, pages 157–177, 2011.
  - [24] Claudio Lucchese, Salvatore Orlando, and Raffaele Perego. A unifying framework for mining approximate top-k binary patterns. *IEEE TKDE*, 26:2900–2913, 2014.
  - [25] Carlos Eduardo Pires, Paulo Sousa, Zoubida Kedad, and Ana Carolina Salgado. Summarizing ontology-based schemas in pdms. In *Data Engineering Workshops (ICDEW), 2010 IEEE 26th International Conference on*, pages 239–244. IEEE, 2010.
  - [26] Alexander Schätzle, Antony Neu, Georg Lausen, and Martin Przyjaciół-Zablocki. Large-scale bisimulation of rdf graphs. In *Proceedings of the Fifth Workshop on Semantic Web Information Management*, page 1. ACM, 2013.
  - [27] Yan Sun, Kongfa Hu, Zhipeng Lu, Li Zhao, and Ling Chen. A graph summarization algorithm based on rdf logistics. *Physics Procedia*, 24:1707–1714, 2012.
  - [28] Yuanyuan Tian, Richard A Hankins, and Jignesh M Patel. Efficient aggregation for graph summarization. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 567–580. ACM, 2008.
  - [29] Yuanyuan Tian and Jignesh M Patel. Interactive graph summarization. In *Link Mining: Models, Algorithms, and Applications*, pages 389–409. Springer, 2010.
  - [30] Zhichun Wang. A semi-supervised learning approach for ontology matching. In *Chinese Semantic Web and Web Science Conference*, pages 17–28. Springer, 2014.
  - [31] Haiwei Zhang, Yuanyuan Duan, Xiaojie Yuan, and Ying Zhang. Assg: adaptive structural summary for rdf graph data. *ISWC*, 2014.
  - [32] Ning Zhang, Yuanyuan Tian, and Jignesh M Patel. Discovery-driven graph summarization. In *Data Engineering (ICDE), 2010 IEEE 26th International Conference on*, pages 880–891. IEEE, 2010.
  - [33] Mussab Zneika, Claudio Lucchese, Dan Vodislav, and Dimitris Kotzinos. Rdf graph summarization based on approximate patterns. In *International Workshop on Information Search, Integration, and Personalization*, pages 69–87. Springer, 2015.
  - [34] Mussab Zneika, Claudio Lucchese, Dan Vodislav, and Dimitris Kotzinos. Summarizing linked data rdf graphs using approximate graph pattern mining. In *EDBT 2016*, pages 684–685, 2016.