Visual Presentation and Summarization of Linked Data Schemas

Lelde Lāce[0000-0001-7650-2355], Aiga Romāne-Ritmane[0009-0003-3609-1485], Mikus Grasmanis[0000-0002-0668-0970], Artūrs Sproģis[0000-0002-2320-0887], Jūlija Ovčiņņikova[0000-0002-5884-763X], Uldis Bojārs[0000-0001-7444-565X], Kārlis Čerāns[1][0000-0002-0154-5294]

Institute of Mathematics and Computer Science, University of Latvia, Riga, Latvia
[1] karlis.cerans@lumii.lv

Abstract. A visually presented schema of a data set can help its user to gain understanding about its contents and use an appropriate vocabulary to build queries or develop applications accessing it. Still, most of Linked Open Data sets are provided without available visual schemas and for data sets of realistic size the schemas may be way too large to be visualized on a suitable diagramming canvas. In this paper we develop a concept of visualization-oriented data schema, involving property ascription point information and propose a method for creating visual presentations of the data schemas, including both the detailed schema visualization and visual schema summarization options. We evaluate the method on 24 prominent data sets from the Linked Open data cloud, where it can obtain legible visualizations of the full schema or at least its fragment.

Keywords: RDF, SPARQL, Linked data, Knowledge graph schema, Visual schema diagram

1 Introduction

Understanding the structure and contents of a Knowledge Graph is important for any user action over it, be that ad hoc data querying or building data consuming applications. The data structure presentation allows seeing if the data set contains the relevant information and in what form (e.g., using what vocabulary) it has been encoded.

A well-known means of data structure description is the data schema that is based on the data set entity (class and property) vocabularies and their connections.

An abstract data schema that involves a class and property mapping or its encoding in RDF data shape language, as SHACL [1] or ShEx [2], can be useful for machine processing, including data validation, user interface form generation or context-aware entity name suggestion e.g., during auto-completion of the text of a query over the data. For human perception a meaningful visual presentation of the data schema would well complement the textual schema presentation as it could invoke the user's visual perception capabilities in schema understanding.

Still, a visual presentation of a data schema in the paradigm of a graph of attributed nodes and edges faces a natural limitation of the size of the graph that can be expected to help the schema perception; often this size shall be less that the number of relevant entities making the data set structure.

To respond to this problem, we investigate the possibilities of presenting the schemas of knowledge graphs in the style of UML class diagrams (representing the classes as nodes and the properties as links or as attributes). We aim at legible presentations of possibly large schemas by developing methods that (i) recognize the relevance of a class as a property source or target and (ii) allow to merge nodes of classes with similar instance incoming and outgoing property characteristics.

We build upon related work that involves on-the-fly computing of schema diagram fragments in LD-VOWL [3] and LODSight [4] (with limitations regarding the schema and the data set size, as well as the details included in the schema), automated means for data schema extraction as (enriched) SHACL shapes (cf. [5]) and abstract class-to-property mappings (cf. [6]), as well as data schema visualization in various notations, involving VOWL [7], OWLGrEd [8], RDFShape [9] and ViziQuer [10].

The main contributions of this paper are:

- describing a mathematical framework for an extended data schema notation, that involves the relevance markers for property ascription at classes,
- developing a method for visual compact data schema representation, allowing to obtain legible summaries for data schemas with well over 100 classes, and
- presenting a library of schema summary or fragment visualizations for 24 prominent data sets from the Linked Open Data[1] cloud.

In what follows, Section 2 provides further Related work details and Section 3 presents the data schema and schema graph concepts. Section 4 then describes basic schema diagrams; Section 5 presents the schema summarization and Section 6 describes the implementation and evaluation. Finally, Section 7 concludes the paper.

The schema visualization methods have been integrated within the open-source ViziQuer tool[2] (where also visual SPARQL query creation is supported). The resources supporting the paper, including the data schema visualizations and the used software startup options are available from https://github.com/LUMII-Syslab/viziquer-tools-lod24. The schema visualization can be performed on ViziQuer Playground[3], as well.

2 Related Work

The visual presentation of relational database schemas is common in most of major database management systems as well as in a variety of custom database handling tools. Still, these tools, as the entire relational database management framework, work on the technical level of tables, columns, and links, therefore they can be considered just as a source of inspiration for the solutions in the knowledge graph and semantic data area.

In the knowledge graph and semantic technology realm there are visual notations and tools for presenting OWL ontologies, such as VOWL [7], OntoDia [11], and

[1] https://lod-cloud.net/

[2] https://github.com/LUMII-Syslab/viziquer

[3] https://viziquer.app

OWLGrEd [8] (cf. also [12] for an overview of ontology visualization methods). Visual ontology editors such as ChOWLk [13] and OWLGrEd allow visual ontology authoring, as well. There are also visualization tools for RDF data shape notations SHACL and ShEx, including, e.g., RDFShape [9], Shacl Play! [14] and shacl2plantUML [15]. To obtain the visualization of the actual data schema using some of these tools, the data schema would need to be described in the respective notation first.

The methods for automated extraction of data shapes from a data set have been attracting research recently, as well, including [5], where a method of extracting an enriched SHACL description of the data set structure from the data dump in .n3 format has been provided. Still, the existing tools for SHACL extraction have not yet been evaluated together with the tools for SHACL visualization. It is also not clear how the SHACL specifications would encode the subclass relation essential for compact schema presentation in the UML class diagram form (e.g., so that a property does not get ascribed both to a super-class and a subclass).

The idea of automatically extracting the schema from an RDF data set has already been explored in [16], where the schema presentation in the form of a UML-style diagram is considered and compact schema presentations are discussed, as well. Notably, this work discusses the need to mine the subclass relation from the data set, although the implementation yielding mixed results. The options for creating the summary diagrams also seem limited to including one "most popular" or "most distinctive" data and object property for a class in the summary diagram, leaving open a desire for obtaining summary diagrams with richer contents.

A recent prominent RDF data summarization tool is RDFQuotient [17] that allows computing the data summary nodes and their relations just from the data set contents, paying attention also to the data resource class information. Although the tool would have an option of re-locating a property ascription from a subclass to a superclass, the possibilities of its finer-grained interaction with the abstraction possibilities and custom class node merging are less clear and the available examples of legible created data structure diagrams are quite size-limited.

Some of early end-user tools allowing visualization of existing Linked Data set schemas involve LD-VOWL [3] and LODSight [4]. These approaches attempt to create the visual data set structure on-the-fly, as the user starts to look at the upcoming diagram (which is quite admirable). Still, this imposes limitations on the size of the data schemas that can be analysed and on the details that can be included with each of the obtained data schemas.

The process of extracting a data schema from the data set by means of a series of SPARQL queries has been conceptually outlined in an earlier work by the authors [6], as well, where the schema had been made available for a visual query environment. An initial experiment of visualizing smaller-scale data set schemas (with up to 50 classes) using a limited schema compacting approach and an external diagram visualization module has been described in [10]. This work has been substantially expanded here in terms of mathematical precision, refinement of compacting methods, and expanding their reach to diagrams of schemas with well over 100 classes, simplified architecture (web-based schema diagram management), and presenting a library of 24 large data sets from the Linked Open Data cloud.

3 Data Schema Graphs

We start the description of the knowledge graph (KG) schema, and the schema graph concepts by introducing the used notation to describe the knowledge graphs themselves.

A ***knowledge graph*** $K=(R,L,C,P,rc,T)$ consists of a set of resources R (involving both external resources Re (corresponding to IRIs) and internal resources Ri (corresponding to blank nodes)), a set of literals L, a set of classes C, a set of predicates P (we allow P to overlap with Re), a resource class assignment $rc: R->2^C$ and a set of triples $T \subseteq R \times P \times (R \cup L)$; we write $(x\ a\ c)$ for $c \in rc(x)$ and simply (x,y,z) for $(x,y,z) \in T$.

If not specified otherwise, we shall assume that the knowledge graph K is given throughout the rest of this section and shall refer to its components by the notation introduced in the previous paragraph.

Given the KG K, let its schema $S(K)$ be $(C,P,cp,pc,cpc,cc,\#_C,\#_P,\#_{cp}.\#_{pc}.\#_{cpc})$, where

- C is the set of K ***classes***, P is the set of K ***properties***, $\#_C(c)=\#\{x|(x\ a\ c)\}$ (# is the set size) and $\#_P(p)=\#\{(x,y)|(x,p,y)\}$ for $c \in C$ and $p \in P$,
- $cp \subseteq C \times P$ and $pc \subseteq C \times P$ are sets of ***class-to-property*** and ***property-to-class links*** and $cpc \subseteq C \times P \times C$ is the set of ***class-property-class links*** such that:
  - $\#_{cp}(c,p)=\#\{(x,y)|(x\ a\ c) \wedge (x,p,y)\}$ and $(c,p) \in cp \Leftrightarrow \#_{cp}(c,p) > 0$,
  - $\#_{pc}(p,c)=\#\{(x,y)|(x\ a\ c) \wedge (y,p,x)\}$ and $(p,c) \in pc \Leftrightarrow \#_{pc}(p,c) > 0$, and
  - $\#_{cpc}(c1,p,c2)=\#\{(x,y)|(x\ a\ c1) \wedge (x,p,y) \wedge (y\ a\ c2)\}$ and $(c1,p,c2) \in cpc \Leftrightarrow \#_{cpc}(c1,p,c2) > 0$.
- $cc \subseteq C \times C$ is the sublass relation (the set of ***subclass-to-superclass pairs***), i.e. $(c1,c2) \in cc$ whenever $(x\ a\ c1) \Rightarrow (x\ a\ c2)$ for all x.

We write $(c_1,c_2) \in cc+$, if c_1 is a strict transitive subclass of c_2, and $c_1 \sim c_2$, if $c_1=c_2$, $(c_1,c_2) \in cc+$ or $(c_2,c_1) \in cc+$ (i.e., c_1 is a (transitive) subclass or superclass of c_2).

For a data schema we define its ***projection*** to (C',P',cc'), where $C' \subseteq C$, $P' \subseteq P$ and $cc' \subseteq cc$ by restricting also the cp, pc and cpc components accordingly (the class restriction induces minimal necessary restriction on cc, as well). The schema projection concept provides a well-defined meaning to ***fragments*** of the full KG schema if the class and property sets are restricted; the restriction of subclass relation (besides the restriction due to the smaller class set) allows working with schema variants with limited information about the subclass relation.

There is a ***natural presentation*** of the ***schema*** in a form of an ***attributed graph*** with the schema classes as nodes and the cpc relation as property-labelled links among classes, while the cp and pc relations are encoded in node attributes designated to hold lists of properties outgoing from and incoming into the respective classes. The cc relation shall be encoded by dedicated subclass links among the nodes in the schema graph. The respective size information can be added to the graph using further decorations.

Drawing the full schema graph (in the presence of a non-empty subclass relation) may quickly indicate an overload of connections, e.g., due to ascribing properties both to the superclasses and subclasses. Figure 1 contains an illustration of a fragment of a simple Nobel Prizes data schema (cf. Figure 2 for the full schema and source credit), including the full class and property connections (left) and only the "essential" ones (right).

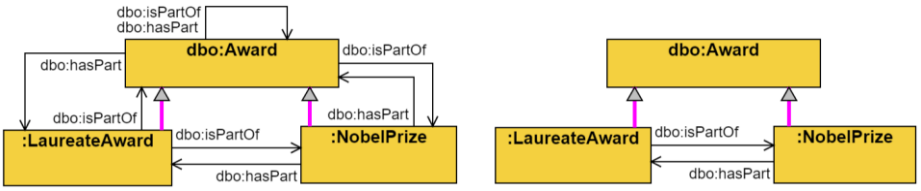

Figure 1. Full (left) and essential (right) class and property connection example

To prepare for reduced schema graph presentation we first define the markers for determining the relevance of a class in the property source or target context.

Given a property p, a class set $A \subseteq C$ is a ***principal source class set*** for p, if

- for any class c and resource x, $(x\ a\ c) \wedge (x,p,y)$ implies $d \in A$ for some $d \sim c$ such that $(x\ a\ d)$ (***coverage condition***),
- if $c_1 \in A$ and $c_2 \in A$ such that $c_1 \neq c_2$, then $\neg(c_1 \sim c_2)$ (***minimality condition***: no property is ascribed both to a subclass and a superclass), and
- if $(x,p,y) \Rightarrow (x\ a\ c)$ (i.e., c is (ontological) domain for p) and $(c,d) \in cc^+$, then $d \notin A$ (***specificity condition***: no property is ascribed to a superclass if it suffices to ascribe it to a subclass).

The principal source class set is the set of classes that characterizes the property presence at the schema class instances: if a property p is present at instances of some class c, then either p is ascribed to the class c itself or to some its superclass, or all instances of c that possess the property p are split among the subclasses of c that each has the property p ascribed to it. On the other hand, the minimality and specificity conditions guarantee that the property ascription is not too verbose (having unnecessary property ascription points) or too generic.

We note that in general the principal source class set for a property can be non-unique, e.g., if $c_i \subseteq c$ for $i=1,2,3$, and the property p applies to instances of c_1 and c_2, but not to instances of c_3, then either the ascription of p to c, or the ascription of p to both c_1 and c_2 would allow to obtain a principal source class set.

We define a ***principal target class set*** B for p in a similar (dual) way:

- $(y\ a\ c) \wedge (x,p,y)$ implies $d \in B$ for some $d \sim c$ such that $(y\ a\ d)$ (coverage),
- if $c_1 \in B$ and $c_2 \in B$ such that $c_1 \neq c_2$, then $\neg(c_1 \sim c_2)$ (minimality), and
- if $(x,p,y) \Rightarrow (y\ a\ c)$, and $(c,d) \in cc^+$ then $d \notin A$ (specificity).

Further on, the principal target and source class sets are similarly defined for a property in its "other end" class context.

For a property p and its source class c a class set B is a principal target class set, if:

- $(x\ a\ c) \wedge (x,p,y) \wedge (y\ a\ c')$ implies $d \in B$ for some $d \sim c'$ s.t. $(y\ a\ d)$ (coverage),
- if $c_1 \in B$ and $c_2 \in B$ such that $c_1 \neq c_2$, then $\neg(c_1 \sim c_2)$ (minimality), and
- if $((x\ a\ c) \wedge (x,p,y)) \Rightarrow (y\ a\ c')$ and $(c',d) \in cc^+$, then $d \notin A$ (specificity).

For a property p and its target class c a principal source class set is defined similarly.

The non-uniqueness observation applies to the principal target class sets for a property, and principal source and target class sets for a property in a context, as well.

Although the concepts of the principal source and target class sets for a property are defined on the level of the KG; one can compute some such principal sets just from the information of the KG schema (the frequency/count information # is essential here).

To compute for a property p a principal source class set, denoted by $PS(p)$, let $src(p)=c_1,c_2,..,c_k$ be the sequence of all p source classes c (having $(c,p) \in cp$), ordered by their triple count $\#_{cp}(c_i,p)$ descending, then (if the triple counts are equal) by the class size $\#_c(c_i)$ ascending (a smaller class (e.g., a subclass) comes before a larger class). Consider then all classes $c_1,c_2,..,c_k$ from $src(p)$ in the index ascension order and include c_i in $PS(p)$ if no $c' \sim c_i$ (i.e., no subclass or superclass of c_i) is already in $PS(p)$.

The computed set $PS(p)$ is a principal source class set for p:

- If $(x\ a\ c)$ and (x,p,y) then c is in $src(p)$ and either $c \in PS(p)$, or c is not included in $PS(p)$ because of some c' $\in PS(p)$ such that $c \sim c'$. This establishes the coverage condition.
- The minimality follows from not including any c into $PS(p)$, if there already is some $c' \in PS(p)$ such that $c \sim c'$.
- Regarding the specificity, let $(x,p,y) \Rightarrow (x\ a\ c)$ and $(c,d) \in cc+$. Let $n=\#_{cp}(c,p)$. By $(x,p,y) \Rightarrow (x\ a\ c)$ we have $\#_{cp}(c',p) \leq n$ for any c'.
  Since c is in $src(p)$, then either (i) $c \in PS(p)$, or (ii) $c_2 \in PS(p)$ for some c_0 such that $c_0 \neq c$, $c_0 \sim c$ and c_0 is before c in $src(p)$.
  Since $\#_{cp}(c_0,p) \leq n=\#_{cp}(c,p)$, the ordering of $src(p)$ entails $\#_{cp}(c_0,p)=n$ and $\#_C(c_0) \leq \#_C(c)$, what excludes $(c,c_0) \in cc+$, so $(c_0,c) \in cc+$.
  So, in either case, $(c^*,d) \in cc+$ for some $c^* \in PS(p)$ such that c^* is before d in $src(p)$ (c^* is either c, if $c \in PS(p)$, or c_0 otherwise); this implies $d \notin PS(p)$, qed.

Note. If the specificity condition would be dropped, a simple way of computing the (simplified) principal source and target class sets would be just to consider the maximal classes (according to the subclass relation) in the respective source/target class sets. In the example of Figure 1, this would lead to all properties ascribed to *:Award*, so missing important details of, e.g., *:isPartOf* connecting just *:LaureateAward* to *:NobelPrize*.

Although the provided algorithm computes the principal source/target class sets for properties just from the data schema, we propose to work with enriched data schemas, where the principal class sets for properties are pre-computed. The presence of the data set itself in the process of the principal class set computation provides options for varying the algorithm to include more specific principal classes, e.g., when a property can be ascribed to several, but not all, subclasses of a given general class. Since the subclass relation computation can be resource-demanding, the schema enrichments built over a weaker version of the coverage condition that does not rely on the subclass relation can be considered, as well, if the data set is available during the enrichment process.

We also extend the schema information with cardinalities, that can be specified either for a property in general, or for a property in a class context. The property domain and range information (in the exclusive, ontological sense) can be ascribed to the properties, as well. Such information, if available, can be further on included in the schema diagram so making it more informative to its users (cf. Section 4)[4].

[4] Further schema constructs may involve, for instance, property adjacency relations (e.g., what properties can "follow" a given property, or what properties can have common subjects or common objects); this information can be important, e.g., if the schema is used to support autocompletion of queries over the data set. We do not focus on these aspects here, as they are less relevant to the current task of constructing schema presentation.

4 Basic Schema Diagrams

A schema diagram can be created from an extended data schema (with principality markers and, optionally, the cardinality and domain and range information, included) by depicting the schema classes as diagram nodes and showing the properties ascribed to a node either as node attributes or as links outgoing from a node. We shall have a property ascribed (in the attribute and/or the link form) to its principal source classes.

For each pair *(c,p)*, where *c* is in the ***principal source class set*** of *p* we shall draw an ***edge*** from the node corresponding to *c*, labelled by the property *p*, to each class *c'* in the ***principal target class set*** of *p* ***in the context*** of the source class *c*.

Note the asymmetry of the source and target ends of the class connections by the properties; this allows to obtain well-defined semantics of the created schema diagrams in the chosen schema visualization approach.

A property *p* ascribed to a class *c* is depicted as an ***attribute*** of the *c* node, if it ***cannot be established*** that all triples *(x,p,y)* for *(x a c)* are covered by some edge outgoing from the *c* node (i.e., for all *y*, such that *(x a c)* and *(x,p,y)*, we have *(y a c_1)* for some c_1 with a *p*-labelled edge from *c* into it); skipping the attribute form in the diagram can be done, e.g., if $\#_{cpc}(c,p,c')=\#_{cp}(c,p)$ for some *c'*.

Figure 2 contains a simple schema of the Nobel Prizes data set[5] (for better presentation an auto-introduced extension with an abstract superclass *dbo:City or dbo:Country* is allowed, that collects the attributes and links common to both subclasses). We draw the subclass-to-superclass relation by dedicated subclass edges in the schema diagram.

The frequency information associated with classes ($\#_c$), property availability at classes ($\#_{cp}$) and links ($\#_{cpc}$) are included in the diagram, as well. Further on, there are cardinalities (e.g., *[1]* and *[*]*) for properties in the context of the respective source class (in the case of an attribute presentation) and in the context of the source and target pair (in the case of a link labelled by a property), as well as domain (marked by *D*) and range (marked by *R*) indications telling if the property appearance place corresponds to the property domain or range (in the ontological, exclusive sense).

The attribute presentation of a property includes options for describing of the "other end" of the property links; in Figure 2 such descriptions involve "IRI" (the property triple objects in the context can be IRIs) and "dgr" (some of the property triples correspond to links drawn in the diagram).

Note that the properties *dct:isPartOf* at *:LaureateAward* class and *:nobelPrize* at *:Laureate* are presented both in the link and the attribute forms since not all property triples in the data have objects that belong to *:NobelPrize*.

The mark ~ at the frequency of *dbo:City or dbo:Country* indicates that the size has been estimated (in this case – calculated as the sum of the subclass sizes, not taking into account the possibility of the subclass overlapping)[6].

---

[5] The data were originally retrieved as a snapshot from https://data.nobelprize.org/sparql and are available on the paper's support page.

[6] Fine-tuning the diagram by calculating the exact sizes of the aggregated items can, in principle, be done by consulting the data set itself during the schema diagram creation.

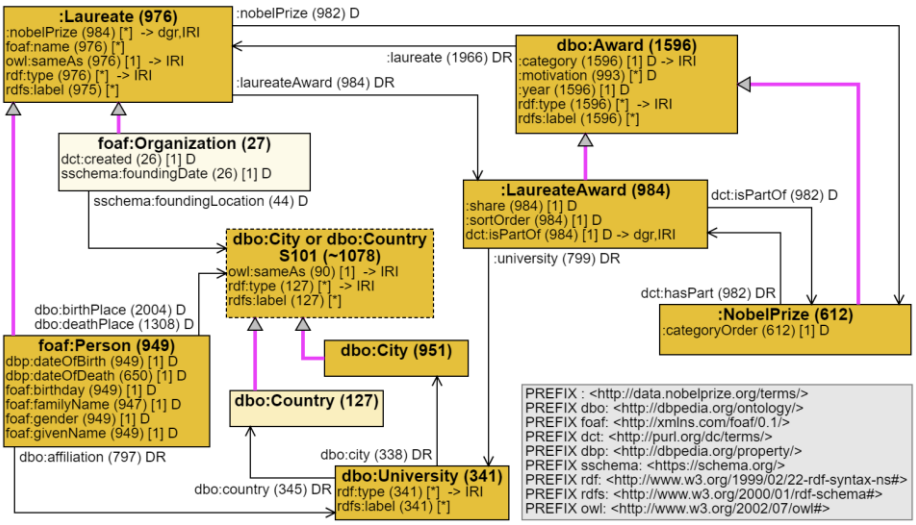

Figure 2. A Nobel Prizes data schema

5 Advanced Schema Summarization

The described schema visualization method works well just for schemas of limited size, as visual diagrams with larger node sets and more involved interlinking patterns can quickly grow beyond easy comprehensibility. To enable the handling of larger schemas, we introduce the following schema summarization and visual tuning methods:

(i) ***class node merging*** (ascribe several classes to a single node; join their attribute and link end lists),

(ii) introducing ***abstract super-classes***, and

(iii) ***inlining links into attributes*** (auto-inlining of circular links into dedicated "looping" attributes and "splitting" certain links into their presentation at the source node (by "outgoing" attributes) and the target node (by dedicated "incoming" attributes).

The parameters of the diagram creation can instruct not to do class node merging at all or to merge just the classes with identical attribute and incoming and outgoing property lists. Aside from these basic cases, the class node merging is performed based on the following principles:

1. For each pair of classes *(A,B)* their ***similarity measure*** *s(A,B)* and ***difference measure*** *d(A,B)* are calculated, as follows:
   a. $s(A,B) = \sum sqrt(min(\#(p,A)/\#_C(A),1) * (min(\#(p,B)/\#_C(B),1))* w(p)$ over all properties *p* incoming into or outgoing out from both *A* and *B*. Here $\#(p,X)$ is $\#_{pc}(p,X)$ for *p* incoming into *X* and $\#_{cp}(X,p)$ for *p* outgoing from *X* (we consider all *A* and *B* properties here, not only those ascribed to them in the schema graph). $w(p)$ is the weight assigned to the property *p* in the similarity computation (usually $w(p)=1$; some properties as *rdfs:label* or

owl:sameAs can be excluded from similarity computation by setting *w(p)=0*).

b. $d(A,B) = \sum sqrt((min(\#(p,A)/\#_C(A),1))^2 * (\#_C(A)/(\#_C(A)+\#_C(B)))$ for all properties (incoming or outgoing) present for the class *A* and not present for the class *B* (the sum is obtained by considering both classes in both roles).

The intuition is that properties belonging to both classes contribute to their similarity, while the properties belonging to one class and not to the other contribute to their difference. A larger property is expected to make a larger contribution.

2. Given a ***size factor*** $ex \geq 0$ (typically also $ex < 1$, sample values are *0, 1/100, 1/10, 1/5* and *1/3*) let the ***ex-weighted similarity and difference measures*** *sw* and *dw* be as follows (*log* is decimal logarithm):

a. $sw(A,B) = s(A,B) / pow((log(cA)+1)*(log(cB)+1),ex)$

b. $dw(A,B) = d(A,B) * pow((log(cA)+1)*(log(cB)+1),ex)$

The intuition is that smaller classes with the same similarity/difference characteristics shall have higher weighted similarity and lower weighted difference and, so, might be used in merging faster.

The similarity and difference measures are generalized to groups of classes, as well. We consider a simple approximate implementation, where the size of the and property appearance in the group is estimated as the sum of the sizes over all group members (this is precise, if classes do not overlap; otherwise, an overestimation is possible).

A merged schema then can, in principle, be defined based on clustering of classes in accordance to some distance function *f(A,B)*, as e.g., $dw(A,B)/(sw(A,B)+\varepsilon)$ for a small ε (some normalization of the function *f* to make it Euclidean may be necessary).

Our approach, alternatively, gives the user explicit control over the conditions when the class nodes can be merged, so that the appearance of class names in a single node can be interpreted as a certain similarity level between these classes (this way we cannot have *a priori* guaranties on size of the obtained schema, though).

We introduce a ***difference threshold*** *G* (typical values are *0,2,5,10* and *20*) and consider a pair of classes *(A,B)*:

- ***similar***, if both $dw(A,B) < s(A,B)$ and $dw(A,B)<G$,
- ***neutral***, if $dw(A,B) < s(A,B)$ and not $dw(A,B)<G$, and
- ***different***, if not $dw(A,B) < s(A,B)$.

We incrementally build (locally) maximal clusters of classes so that:

- all pairs of classes in a cluster are either similar or neutral, and
- for each class in a cluster at least 50% of the other classes are similar to it.

Should the size of the created clustered data set be above the desired target, another try with changed *ex* and *G* parameters can be done, or a fragment of the schema (in terms of classes and/or properties), e.g., the largest ones, can be considered. Several diagrams, based on different property sets, can be created, as well.

The ***introduction of abstract super-classes***, if requested, is based just on the similarity of the classes (or groups of classes) and the properties that are present in at least two subclasses of the abstract superclass are brought up to the super-class level. This is a finer diagram abstraction mechanism if compared to the class node merging, as it allows the distinct properties (attribute and link ends) at each subclass to stay there and not be brought up to the superclass level. Still, it comes at a cost of introducing an extra node in the diagram (instead of merging several nodes into one), and its introduction

needs to be justified mainly by simplifying the overall link structure in the diagram (note that the introduction of the *dbo:City or dbo:Country* abstract class in Figure 2 allowed to merge the links both from *foaf:Person* and *foaf:Organization*, as well as the entire attribute list from both subclasses).

There are two types of conditions that induce the ***inlining of a property link*** into the nodes of its ends (at each end keeping the information of the other property ends):

- maximum number of same-property edges in a diagram (e.g., 7), and
- a minimum number of triples for a property to be shown as line is set.

Figure 3 contains a presentation of an example schema of *StarWars* data set [18], obtained using the described schema summarization method (there are 51 classes in the data set, summarized into 13 nodes (the node *Wookie et.al.* is obtained as summarization of 36 classes); the presentation of the diagram in the tool allows seeing the class and property list elements that are not included in the visual presentation, as well).

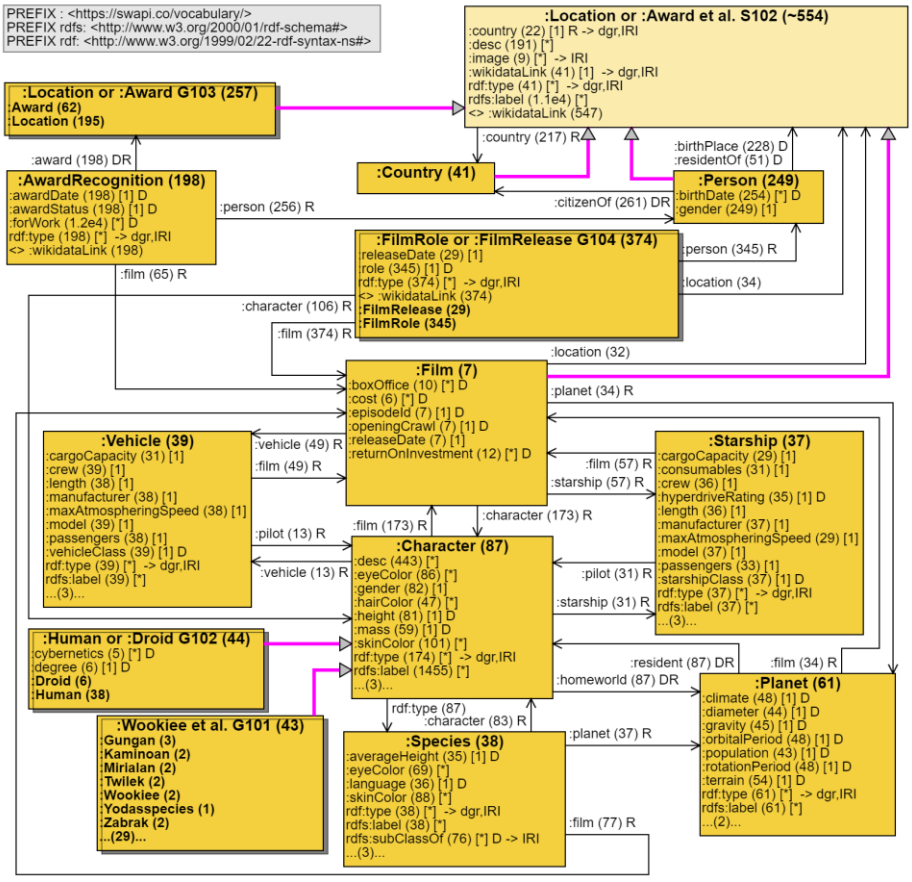

Figure 3. StarWars data schema example

6 Implementation and Evaluation

The visual schema presentation assumes the availability of an (enriched) data schema, as described in Section 3. To enable work with schemas of realistic size, we have the schemas pre-computed. We use the open-source OBIS Schema Extractor tool[7] that retrieves the schema from a SPARQL endpoint. The schemas are stored using the Data Shape Server (DSS) tool[8] (both schema storage and schema serving functionality included) and then are seamlessly accessed from the ViziQuer tool environment, where both the schema visualization and schema-based visual queries (cf. [19]) are available. The ***links to the live examples*** of the considered schemas on ViziQuer playground, as well as a ***Docker-based environment*** for setting up and running the examples locally are provided on the paper support resource.

The schema visualization is initiated by the 'Data Schema' button in the ViziQuer environment project (diagram list) view, after what the window with parameter setup is opened. The parameters to be tuned involve the list of classes and properties to be visualized (there are sliders available for choosing the largest classes and/or properties, as well as manual options for selecting classes and properties) and the diagram merging parameters: ***merging strength*** (difference threshold in Section 5), ***size factor*** (cf. Section 5) and link inlining parameters (***number of same-property lines*** and ***triple count threshold***). After the parameters have been set, there is an option (via the button 'Show merged classes') to see the counts of nodes and links, as well as the contents of nodes that are going to be created in the diagram. At this point the parameters can be adjusted to ensure that the created diagram is going to be of reasonable size. As a rule of thumb, it would not be recommended to draw diagrams with more than 100 nodes[9].

The button 'Create Schema diagram' creates the diagram and places a pointer to it in the project diagram view. When opening the diagram for the first time, it is automatically laid out (it may take a little time), after what it can be observed and manually tuned by moving the diagram nodes around. For smaller diagrams (around 20 nodes), the tuning is going to be rather easy, while for larger diagrams it gets more complicated. If the node count approaches 100, or even if it is about 40 – 60 with a complex line configuration, it can take a couple of hours for a professional to produce a reasonably well looking diagram (if the diagram tuning task appears too difficult, another schema diagram of smaller size (e.g., for a fragment of the schema, or obtained via stronger compacting parameters) can be created and worked with).

We note that if the property list at a node is too large to be shown in full in the node box, the full property list can be seen in the side panel, if selecting the respective node.

To evaluate the schema visualization method, we consider 24 prominent Linked Open Data sets registered in the Linked Open Data cloud[10]. For specificity, we consider the data sets meeting the following criteria:

---

[7] https://github.com/LUMII-Syslab/OBIS-SchemaExtractor

[8] https://github.com/LUMII-Syslab/data-shape-server

[9] Node count above 150 can cause temporal freezing of the system in the current implementation

[10] https://lod-cloud.net/lod-data.json

- the data set is available via a SPARQL endpoint (only 145 out of 1584 data sets had a responsive SPARQL endpoint at the time of the experiment),
- the data set is not a part of multiple data set agglomeration on the same SPARQL endpoint[11] (so the data access information can be uniformly retrieved from the LOD cloud data file *sparql* section that does not list named graphs),
- the endpoint triple count, list of classes (with instance count), and the list of properties (with triple count) are accessible via direct aggregated SPARQL queries, and the schema extractor produces a valid data schema[12],
- the (full) class count is between 15 and 2000, and the property count is below 10000 (the smaller schemas are easy and are well-handled, e.g., in [10]; we do not pretend to show diagrams of too heterogenous data sets either).

We order the obtained list in descending order by the actual triple count. For the evaluation, we take the top 21 of the obtained data sets, add a custom *AcademySampo* practical data set, and two example data sets (with a reasonably large triple count) from the *foodie cloud* aggregation, resulting in 24 data sets for which we create the diagram visualizations (either the full diagrams, or the fragments with the largest classes). The diagram visualization experience is summarized in Table 1. The columns *Classes* and *Properties* may contain two numbers each, indicating the full class/property count and the relevant class/property count (excluding, e.g., the *OpenLink Virtuoso* system classes and properties). In the column Lines the numbers are given both for drawing lines of all sizes (excluding just the same-property lines, if more than 7), and for drawing lines corresponding to at least 100 triples). The node and line counts, if not stated otherwise, are given for the maximum strength merging (strong merging (20) and size factor 0); if the diagram size and appearance permits, larger and semantically more nuanced diagrams can be obtained by lowering the merging strength and raising the size factor.

The considered data schemas are available for experimenting on the ViziQuer playground (e.g., creating schema diagrams using various parameters) or by running the ViziQuer tools software locally. The created schema diagrams are also available both in the image form and within a project that can be loaded into the visual tool.

Table 1. SPARQL endpoint schema presentation experiment

Endpoint URL	Triples	Classes	Properties	Nodes	Lines	Notes
https://libris.kb.se/sparql	2695020210	540	939	225	467(0) 417(100)	A fragment with 40 largest classes can be shown as a diagram with 22 nodes (larger diagrams are more difficult due to high line count).
https://sparql.nextprot.org/	2130123293	165	238	39	66(0) 65(100)	OK
http://affymetrix.bio2rdf.org/sparql	1377023559	661	1328	132	307(0) 271(100)	A fragment with 100 largest classes can be shown as a diagram with 45 nodes.

[11] This excludes, e.g., the data sets from https://www.foodie-cloud.org/sparql (two data sets are brought back into the experiment manually), http://publications.europa.eu/webapi/rdf/sparql, https://linked.opendata.cz/sparql and http://opendatacommunities.org/sparql

[12] In the case of the SPARQL endpoints considered here, the extractor had been able to produce a valid schema in all but one cases when the class and property lists were possible to obtain.

https://ruian.linked.opendata.cz/sparql	870638775	85	200	43	53(0) 26(100)	OK
http://data.bnf.fr/sparql	651506623	38 (26)	886	19	39(0)	Basic merging, OK.
http://kaiko.getalp.org/sparql	522998164	140 (128)	332 (245)	31	34(0) 28(100)	OK (size factor 1/5 or 1/3 recommended).
http://cr.eionet.europa.eu/sparql	482077457	272	2001	220	350(0) 308(100)	A fragment with 50 largest classes can be shown as a diagram with 46 nodes.
http://dati.isprambiente.it/sparql	385222839	135 (122)	383	86	135(0) 101(100)	Legible diagram (large); for presentation purposes fragments can be considered. OK.
http://dati.camera.it/sparql	322885735	104 (92)	367 (283)	61	119(0) 113(100)	OK
http://data.allie.dbcls.jp/sparql	287461727	55 (43)	201	28	37(0)	OK.
http://datos.bne.es/sparql	258140051	28 (16)	329	15	32(0)	Basic merging, OK (externally fetched labels)[13]
http://rdf.disgenet.org/sparql/	99381703	122 (110)	665	46	78(0) 48(100)	OK.
https://taxref.mnhn.fr/sparql	82745498	1925 (1913)	697 (608)	58	96(0) 73(100)	OK. (Note: very large namespace list).
http://opendata.aragon.es/sparql	70049160	218 (206)	1355 (1259)	88	119(0) 103(100)	OK.
Muziekweb[14]	37114240	30	59	16	20(0) 20(100)	OK.
http://premon.fbk.eu/sparql	32611819	123 (95)	234 (146)	46	42(0) 36(100)	OK.
http://geo.linkeddata.es/sparql	29884998	360	212	41	26(0) 12(100)	OK (textual form of the class names maintained, as in the data set).
http://en.openei.org/sparql	27317782	1612 (1600)	5163	306	209(0) 81(100)	A fragment with 100 largest classes can be shown as a diagram with 37 nodes.
http://datos.bcn.ck/sparql	52057935	542	357	93	128(0) 115(100)	Large, yet legible diagram. Fragments recommended for the first impression. OK.
http://id.eaufrance.fr/sparql	17743557	88 (76)	420 (318)	56	58(0) 46(100)	OK.
http://ldf.fi/warsa/sparql	14385118	90	310	57	110(0) 96(100)	OK
http://ldf.fi/yoma/sparql	6627922	267	123	34	60(0) 53(100)	OK
poi.rdf[15]	410867958	290	46	5 41[16]	1(0)	OK (subclass lines not counted)
catchrecords Norway[17]	192867166	18	49	17	17(0) 17(100)	Basic merging, OK.

The performed experiment allows to make the following observations:

[13] The labels were not present in the dataset and not all entities had labels provided in a structured form at the definition page https://datos.bne.es/def/index-en.html

[14] https://data.muziekweb.nl/MuziekwebOrganization/Muziekweb/sparql/Muziekweb

[15] https://www.foodie-cloud.org/sparql, Named graph: http://www.sdi4apps.eu/poi.rdf

[16] Merge equivalent classes only.

[17] https://www.foodie-cloud.org/sparql, graph: http://w3id.org/foodie/open/catchrecord/norway/

- For all considered schemas either full or fragment-based well-legible presentations can be created (sample diagrams are available in the diagram library).
- For endpoints with up to 200-250 classes legible summaries of the full data schema can be reasonably expected if strong summarization is applied and if the properties corresponding to smaller triple counts are inlined. Still, the actual visual complexity of the diagrams may vary from one data endpoint to another
- It might be worthwhile to attempt summarization of full data set schemas of larger size, as well (there are successful legible summaries of full schemas for endpoints with 290, 360, 542 and even 1913 classes in the diagram library).
- The light-weight schema merging techniques (e.g., merging of equivalent classes or basic merging with higher penalties for large class merging) can be recommended also for schemas of smaller size, as this would enable both faster schema summary creation and easier comprehension.
- For the first impression, maximum strength summarization can be chosen without introducing higher summarization penalties for larger classes; in the case of large line counts consider property inlining. The fragment-based presentation or considering just a fraction of larger classes can also be beneficial to obtain the first impression.

7 Conclusions

We have developed a mathematically precise concept of an abstract knowledge graph schema based on the knowledge graph data classes, properties and their relations (involving the class-to-property and subclass relations) and including the principal property ascription points that allow presenting the schema diagram while avoiding the non-informative property-to-class attachments (e.g., avoiding the display of a property both at a superclass and at a subclass).

To handle visual presentation of larger schemas that would not fit on a typical diagramming canvas, possible methods for the schema node grouping and property inlining have been presented. An experiment was conducted in creating visual schema diagrams for realistic Linked Open Data sets (with the class limit of 2000 and property limit of 10000), where for 20 out of 24 data sets the summaries of the full data set schema has been legibly visualized, while for the rest 4 data sets the schema fragment visualization has been successfully performed.

The library of the visual schema presentations for the selected data sets has been created and is offered to the community to support the understanding of the structure of the these data sets by their users; the schema diagrams are offered both in the visual image form, as created by the authors of this paper, and in a live form within the ViziQuer tool, where the interested parties can do further tuning, or create alternative versions of diagrams using different visualization parameters.

The anticipated further work on visual schema presentation would involve expanding the library of data schema visualizations, systematic comparison of different schema summarization strategies, and involving potential end users in the evaluation.

Acknowledgments. This work has been supported by the Latvian Science Council Grant lzp-2021/1-0389 “Visual Queries in Distributed Knowledge Graphs”.

References

1. Shapes Constraint Language (SHACL). W3C Recommendation, 20 July 2017. https://www.w3.org/TR/shacl/
2. Shape Expressions Language 2.1. Final Community Group Report, 8 October 2019. https://shex.io/shex-semantics/
3. Weise, M., Lohmann, S., & Haag, F.: Ld-vowl: Extracting and visualizing schema information for linked data. In 2nd International Workshop on Visualization and Interaction for Ontologies and Linked Data, pp. 120-127 (2016).
4. Dudáš, M., Svátek, V., Mynarz, J.: Dataset Summary Visualization with LODSight. In: The Semantic Web: ESWC 2015 Satellite Events. LNCS, vol. 9341 (2015).
5. Rabbani, K., Lissandrini, M., & Hose, K.: Extraction of validating shapes from very large knowledge graphs. In Proceedings of the Very Large Databases 2023, 16(5), pp. 1023–1032.
6. Čerāns, K., Ovčiņņikova, J., Bojārs, U., Grasmanis, M., Lāce, L., Romāne, A.: Schema-Backed Visual Queries over Europeana and Other Linked Data Resources. In Verborgh, R., et al. (ed.), ESWC 2021 Satellite Events. Springer LNCS, vol. 12739, pp. 82–87 (2021). https://doi.org/10.1007/978-3-030-80418-3_15
7. Lohmann, S., Negru, S., Haag F., Ertl, T.: Visualizing Ontologies with VOWL. In: Semantic Web 7(4), pp. 399–419 (2016).
8. Bārzdiņš, J., Čerāns, K., Liepiņš, R., Sproģis, A.: UML Style Graphical Notation and Editor for OWL 2. In: Proc. of BIR’2010, LNBIP, Springer 2010, vol. 64, pp. 102–113 (2010).
9. Labra Gayo, J. E., Fernández-Álvarez, D., & Garcıa-González, H.: RDFShape: An RDF playground based on Shapes. CEUR Workshop Proceedings, vol. 2180 (2018).
10. Lāce. L., Romāne, A., Fedotova, J., Grasmanis, M., Čerāns, K.: A Method and Library for Visual Data Schemas. To appear in Proc. of ESWC’2024 Satellite Events, Springer LNCS (2024).
11. Mouromtsev, D., Pavlov, D., Emelyanov, Y., Morozov, A., Razdyakonov, D., Galkin, M.: The simple, web-based tool for visualization and sharing of semantic data and ontologies. In: ISWC P&D 2015, CEUR, vol.1486, http://ceur-ws.org/Vol-1486/paper_77.pdf, (2015).
12. Dudáš, M., Lohmann, S., Svátek, V., Pavlov, D.: Ontology visualization methods and tools: a survey of the state of the art. In: The Knowledge Engineering Review, 33, (2018)
13. Chávez-Feria, S., García-Castro, R., & Poveda-Villalón, M.: Chowlk: from UML-based ontology conceptualizations to OWL. In European Semantic Web Conference. Springer LNCS, vol. 13261, pp. 338–352 (2022). https://doi.org/10.1007/978-3-031-06981-9_20
14. Draw UML from SHACL. https://shacl-play.sparna.fr/play/draw, (last accessed on 2024-10-24).
15. Shacl2plantuml. https://github.com/rosecky/shacl2plantuml, (last accessed 2024-10-24).
16. Li H., Zhang X.: Visualizing RDF data profile with UML diagram. Springer Proceedings in Complexity, pp. 273 – 285 (2013). https://doi.org/10.1007/978-1-4614-6880-6_24
17. Goasdoué, F., Guzewicz, P., & Manolescu, I.: RDF graph summarization for first-sight structure discovery. The VLDB journal, 29(5), pp. 1191–1218 (2020).
18. Star Wars, Example Dataset. Available at https://platform.ontotext.com/semantic-objects/datasets/star-wars.html (last accessed on 2024-07-05).
19. Čerāns, K., Šostaks, A., Bojārs, U., Ovčiņņikova, J., Lāce, L., Grasmanis, M., Romāne, A., Sproģis, A., Bārzdiņš, J.: ViziQuer: A Web-Based Tool for Visual Diagrammatic Queries Over RDF Data, in ESWC 2018 Satellite Events. ESWC 2018. Springer LNCS, Vol. 11155. pp. 158–163 (2018). https://doi.org/10.1007/978-3-319-98192-5_30