# Dataset Summary Visualization with LODSight

Marek Dudáš(✉), Vojtěch Svátek, and Jindřich Mynarz

University of Economics, Prague, Czech Republic
{marek.dudas,svatek}@vse.cz, mynarzjindrich@gmail.com

**Abstract.** We present a web-based tool that shows a summary of an RDF dataset as a visualization of a graph formed from classes, datatypes and predicates used in the dataset. The visualization should allow to quickly and easily find out what kind of data the dataset contains and its structure. It also shows how vocabularies are used in the dataset.

## 1 Introduction

In contrast to the RDBMS world, (RDF) datasets on the semantic web usually are not provided with a schema. There are of course RDFS/OWL vocabularies,[1] but those only define sets of concepts that *can* be used in a dataset rather than what combinations of concepts *should* be used to describe the data. The relationship between a vocabulary and a dataset is thus by far not as strict as the relationship between an SQL database and its schema. If users encounter an RDF *dataset* they are not familiar with, finding out what kind of data it contains is nontrivial since up-to-date and complete documentation of the dataset is usually not present. The other way around, when users encounter a *vocabulary* they are not familiar with, they can try to learn the proper usage of the vocabulary by reading the documentation and inspecting the axioms, labels and comments in the RDFS/OWL source code. However, even the vocabulary documentation may be insufficient or missing, and the axioms do not fully specify the usage, as said above. Then the only remaining option is to look at datasets where the vocabulary is used (provided they exist).

Users can obviously explore the dataset manually, e.g., using exploratory SPARQL queries. Several approaches to dataset summarization that makes such exploration easier have been implemented (discussed in Sect. 2). We present LODSight:[2] a dataset summary visualization tool based on those existing approaches. LODSight is aimed to be applicable to any RDF dataset available through a SPARQL endpoint and to show the whole dataset summary in one view. It searches the dataset for typical combinations of class instances and properties. The combinations are merged into one graph and displayed as an interactive node-link JavaScript-based visualization. The application is designed to allow a non-expert user to both (a) get an overview of the data and its structure in the dataset, and, (b) learn how the vocabularies are used in it.

[1] By vocabulary we mean ontology or vocabulary.
[2] Available at http://lod2-dev.vse.cz/lodsight.


F. Gandon et al. (Eds.): ESWC 2015, LNCS 9341, pp. 36–40, 2015.
DOI: 10.1007/978-3-319-25639-9_7

## 2 Related Research

The visualization in LODSight is based on the same principle as *maps of ontology usage* [3]. However, maps of ontology usage focus the visualization on entities from single namespace while we display all classes in one summarization regardless of namespace. Maps of ontology usage rely on YARS2 while we support remote summarization of theoretically any SPARQL endpoint. **ExpLOD** [4] offers a more complex approach based on bisimulation contraction. The result is a node-link visualization similar to ours but more accurate: showing a combination of links that reportedly exist in the dataset while we show combinations of links that *possibly* exist. Our visualization might be on the other hand more intuitive as it shows types of instances directly as node labels while ExpLOD shows types as separate nodes which might lead to clutter. In addition to both maps of ontology usage and ExpLOD we allow to show examples of existing combinations of instances represented by the generalized (sub)graph. An approach similar to ExpLOD is presented by Li [5]. None of the above mentioned tools seem to be publicly available. **SPARQture** [6] shows a summary of the dataset in the form of a mere list of classes but allows incremental exploration showing predicates and example instances related to a selected class. Incremental exploration is applicable on datasets of the size of DBpedia, which is simply too big for detailed view in LODSight. Somewhat similar results are offered by **Rhizomer** [1], which includes a method for displaying the hierarchy of classes used in the dataset in a treemap. We use the notion of *type-property paths* as mentioned by Presutti et al. [9] (and also as used in [10]), which aims at construction of prototypical SPARQL queries showing examples of typical data. Campinas et al. [2] discuss efficient graph summarization using MapReduce, but we aim at easier applicability accessing a SPARQL endpoint directly without the need to transform the data. Our approach is also inspired by the notion of characteristic sets [7] as used by Minh-Duc [8] to summarize a dataset into a relational schema.

## 3 Summarization Method

Summarization in LODSight is based on type-property and datatype-property paths. Type-property path of length 1 is a sequence `type1 - property - type2`. `Type1` and `type2` are the types of instances from the dataset that are connected by the property. We use the term path frequency to denote the number of triples `?s ?property ?o` in the dataset where `?s` is an instance of `type1` and `?o` of `type2`. We use the term *datatype-property path* for a sequence `type - data property - datatype`. Such path is present in the dataset if there is a triple `?s ?dataproperty ?l`, where `?s` is an instance of `type` and `?l` is a literal of type `datatype`. To create a dataset summary in LODSight we first find all type-property paths and merge them into one graph. Then we add datatype-property paths, whose subject type is present in the type-property paths.

The generalization is to some extent inaccurate. Each path has its instantiations existing in the dataset – there are instances typed and connected as

the path shows. However, the graph created by merging the paths together may not necessarily have such instantiations in the dataset. This is explained on an example in Sect. 4. The benefit is that the final summarization is more compact, but the user has to interpret it correctly. The summarization algorithm is implemented as a Java application with Apache Jena library performing SPARQL queries on remote endpoints. The first step of the algorithm is done in a single query:

```
SELECT ?a ?p1 ?b (COUNT(*) AS ?count)
WHERE { [ a ?a ] ?p1 [ a ?b ] .} GROUP BY ?a ?p1 ?b
```

The datatype-property paths are then obtained through a series of additional queries. All paths are stored into a MySQL database and served through a script to a JavaScript application which uses D3.js library to show them in a node-link force-directed layout visualization. The summarization is not done on demand as it might be prohibitively time consuming. The JS app offers a list of prepared summarizations from which the user selects.

*Evaluation.* We tested LODSight on first 32 available datasets listed by default order at datahub.io.[3] In 17 cases the summarization worked, out of which in 10 cases only the type-property paths were retrieved, while queries for datatypes ended in errors.[4] Successful (including datatypes) summarizations of datasets with triple counts between 18,834 and 11,485,244 took between 4 and 44 s. Successful summarization of 117,000,000 triples took 5 min and 7 s.

## 4 Demonstration

An example of LODSight visualization is shown in Fig. 1. It displays a part of a public contracts dataset.[5] Each class is represented as one node, regardless of how many type-property paths it occurs in. However, the fact that a class node is in several paths does not automatically mean that there are actual instances linked as the visualization suggest. In Fig. 1 the nodes pc:Tender, gr:BusinessEntity and schema:ContactPoint are interlinked. That does not necessarily mean that each instance of gr:BusinessEntity with schema:ContactPoint specified is participating in some pc:Tender. It only means that there are some instances of pc:Tender linked to some instances of gr:BusinessEntity and some (possibly different) instances of gr:BusinessEntity that are linked to instances of schema:ContactPoint. Paths are represented by links labeled with predicates between the nodes. Technically, each link is split into two edges and the label is drawn as a diamond shaped node to avoid label and edge overlap. If there are multiple paths between two classes, they are drawn as one link with multiple labels to avoid clutter. The thickness of edges represents the frequency of

[3] The list was filtered to show only datasets provided with SPARQL endpoint.
[4] The problem seems to be in support of demanding queries and bind() and datatype() functions in specific endpoints.
[5] Available at http://lod2-dev.vse.cz/lodsight/?sumid=5024128&minfreq=1.

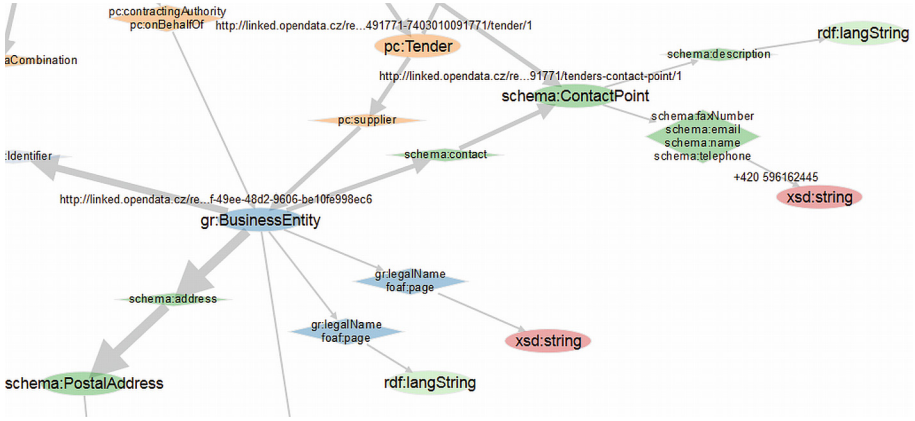


**Fig. 1.** An example of a "zoomed-in" summarization of public contracts dataset.

type-property paths (datatype-property frequency is not counted in the current implementation): in Fig. 1 there are more links from br:BusinessEntity to schema:PostalAddress than schema:ContactPoint. If more classes occur in a datatype-path with the same datatype the datatype node (and the link) is repeated for each such class to avoid edge crossings: see that xsd:string is displayed twice in Fig. 1. After the initial automatic layout is performed the visualization zooms out to show an overview of the whole summarization. The user can pan & zoom and drag & drop the nodes freely then.

Three main features contribute to the goals stated in the introduction: first, each node is colored according to its namespace. Therefore, the user can easily see which parts of the dataset structure are realized with which vocabularies. Second, the user can change the minimum frequency threshold of the type-property paths. A slider is provided that can change the threshold between 1 and a value near the maximum frequency in the dataset. The latter leads to displaying only the most frequent paths thus showing the "core" structure of the dataset. Third, the user can select any subset of the nodes (a subgraph) to retrieve example instantiations of it. The instantiations are retrieved on-line from the SPARQL endpoint and displayed above the corresponding nodes one combination at a time while the user can switch between them: see examples in Fig. 1 showing a combination of instances of pc:Tender, gr:BusinessEntity, schema:ContactPoint and the phone number literal that are actually present in the dataset and linked as the visualization suggests.

The demonstration will include showing visualizations of 17 datasets at various levels of detail and retrieving example instantiations for selected subgraphs.

## 5 Conclusions and Future Work

We presented LODSight: an RDF dataset summary visualization tool that displays typical combinations of types and predicates in a similar way as other existing approaches. There are four main advantages of our approach. First,

it relies solely on SPARQL and thus allows to summarize (theoretically) any dataset accessible through a SPARQL endpoint. Second, it enables to dynamically change the level of detail and show summarizations of very large datasets. Third, it provides an option to load examples of instances and literals represented by a selected subgraph. Last, it makes the visualizations publicly available through a web browser. Real-time summarization might be considered for smaller datasets – preliminary results suggest that processing several hundred triples takes less than 20 s. For large datasets, implementing incremental exploration similar to SPARQture might be an option. We also plan to improve the reliability of the summarization: preliminary evaluation shows it worked on 53 % of tested datasets. A possible workaround for that is copying the dataset into a more reliable triple store and running the summarization there. Future evaluation will include tests with a group of users, students with basic knowledge of linked data, asked to describe displayed data or learn the usage of a vocabulary.

*This research is supported by VŠE IGA No. F4/90/2015 and long-term institutional support of research activities by Faculty of Informatics and Statistics, University of Economics, Prague.*

## References

1. Brunetti, J.M., Garca, R., Auer, S.: From overview to facets and pivoting for interactive exploration of semantic web data. Int. J. Semantic Web Inf. Syst. **9**, 120 (2013). doi:10.4018/jswis.2013010101
2. Campinas, S., et al.: Efficiency and precision trade-offs in graph summary algorithms. In: Proceedings of the 17th International Database Engineering & Applications Symposium, pp. 38–47. ACM (2013)
3. Kinsella, S., et al.: An interactive map of semantic web ontology usage. In: 12th International Conference Information Visualisation, IV 2008, pp. 179–184. IEEE (2008)
4. Khatchadourian, S., Consens, M.P.: ExpLOD: summary-based exploration of interlinking and RDF usage in the linked open data cloud. In: Aroyo, L., Antoniou, G., Hyvönen, E., ten Teije, A., Stuckenschmidt, H., Cabral, L., Tudorache, T. (eds.) ESWC 2010, Part II. LNCS, vol. 6089, pp. 272–287. Springer, Heidelberg (2010)
5. Li, H.: Data profiling for semantic web data. In: Wang, F.L., Lei, J., Gong, Z., Luo, X. (eds.) WISM 2012. LNCS, vol. 7529, pp. 472–479. Springer, Heidelberg (2012)
6. Maali, F.: SPARQture: A More Welcoming Entry to SPARQL Endpoints. http://ceur-ws.org/Vol-1279/iesd14_9.pdf
7. Neumann, T., Moerkotte, G.: Characteristic sets: accurate cardinality estimation for RDF queries with multiple joins. In: 2011 IEEE 27th International Conference on Data Engineering (ICDE), pp. 984–994. IEEE (2011)
8. Pham, M.: Self-organizing structured RDF in MonetDB. In: 2013 IEEE 29th International Conference on Data Engineering Workshops (ICDEW), pp. 310–313. IEEE (2013)
9. Presutti, V., et al.: Extracting core knowledge from linked data. In: Proceedings of the Second Workshop on Consuming Linked Data, COLD 2011 (2011)
10. Svátek, V., et al.: B-Annot: supplying background model annotations for ontology coherence testing. In: 3rd Workshop on Debugging Ontologies and Ontology Mappings at ESWC 2014, Heraklion, Crete (2014)