The Linked Data Visualization Model

Conference Paper · January 2012

CITATIONS
46

READS
1,123

3 authors:

Josep Maria Brunetti
Universitat de Lleida
20 PUBLICATIONS **204** CITATIONS
SEE PROFILE

Sören Auer
Leibniz Universität Hannover
545 PUBLICATIONS **17,208** CITATIONS
SEE PROFILE

Roberto García
Universitat de Lleida
108 PUBLICATIONS **1,377** CITATIONS
SEE PROFILE

The Linked Data Visualization Model

Josep Maria Brunetti[1], Sören Auer[2], Roberto García[1]

[1] GRIHO, Universitat de Lleida
Jaume II, 69. 25001 Lleida, Spain
{josepmbrunetti,rgarcia}@diei.udl.cat, http://griho.udl.cat/
[2] AKSW, Computer Science
University of Leipzig, Germany
auer@informatik.uni-leipzig.de, http://aksw.org/

Abstract. In the last years, the amount of semantic data available in the Web has increased dramatically. The potential of this vast amount of data is enormous but in most cases it is very difficult for users to explore and use this data, especially for those without experience with Semantic Web technologies. Applying information visualization techniques to the Semantic Web helps users to easily explore large amounts of data and interact with them. The main objective of information visualization is to transform and present data into a visual representation, in such a way that users can obtain a better understanding of the data. However, the large amount of Linked Data available and its heterogeneity makes it difficult to find the right visualization for a given dataset. In this article we devise a formal Linked Data Visualization model (LDVM), which allows to *dynamically* connect data with visualizations. In order to achieve such flexibility and a high degree of automation the LDVM is based on a visualization workflow incorporating analytical extraction and visual abstraction steps. We report about our comprehensive implementation of the Linked Data Visualization model comprising a library of generic visualizations that enable to get an overview on, visualize and explore the Data Web.

Keywords: Semantic Web, Linked Data, Visualization, Interaction

1 Introduction

In the last years, the amount of semantic data available in the Web has increased dramatically, especially thanks to initiatives like Linked Open Data (LOD). The potential of this vast amount of data is enormous but in most cases it is very difficult and cumbersome for users to visualize, explore and use this data, especially for lay-users [1] without experience with Semantic Web technologies.

Visualizing and interacting with Linked Data is an issue that has been recognized from the beginning of the Semantic Web (cf. e.g. [2]). Applying information visualization techniques to the Semantic Web helps users to explore large amounts of data and interact with them. The main objectives of information visualization are to transform and present data into a visual representation, in

such a way that users can obtain a better understanding of the data [3]. Visualizations are useful for obtaining an overview of the datasets, their main types, properties and the relationships between them.

Compared to prior information visualization strategies, we have a unique opportunity on the Data Web. The unified RDF data model being prevalent on the Data Web enables us to bind data to visualizations in an *unforeseen* and *dynamic* way. An information visualization technique requires certain data structures to be present. When we can derive and generate these data structures automatically from reused vocabularies or semantic representations, we are able to realize a largely automatic visualization workflow. Ultimately, we aim to realize an ecosystem of data extractions and visualizations, which can be bound together in a dynamic and unforeseen way. This will enable users to explore datasets even if the publisher of the data does not provide any exploration or visualization means.

Most of existing work related with visualizing RDF is focused on concrete domains and concrete datatypes. However, finding the right visualization for a given dataset is not an easy task [4]: *'One must determine which questions to ask, identify the appropriate data, and select effective visual encodings to map data values to graphical features such as position, size, shape, and color. The challenge is that for any given data set the number of visual encodings – and thus the space of possible visualization designs – is extremely large.'*

The Linked Data Visualization Model we propose in this paper allows to connect different datasets with different visualizations in a dynamic way. In order to achieve such flexibility and a high degree of automation the LDVM is based on a visualization workflow incorporating analytical extraction and visual abstraction steps. Each of the visualization workflow steps comprises a number of transformation operators, which can be defined in a declarative way. As a result, the LDVM balances between flexibility of visualization options and minimality of configuration.

Our main contributions are in particular:

1. The adoption of the Data State Reference Model [5] for the RDF data model.
2. The creation of a formal Linked Data Visualization model, which allows to *dynamically* connect data with visualizations.
3. A comprehensive implementation of the Linked Data Visualization model comprising a library of generic visualizations that enable to get an overview on, visualize and explore the Data Web.

The remainder of this paper is structured as follows: Section 2 discusses related work. Section 3 introduces the Linked Data Visualization Model. Section 4 describes an implementation of the model and Section 5 presents its evaluation with different datasets and visualizations. Finally, Section 6 contains conclusions and future work.

2 Related Work

Related work can be roughly classified into tools supporting Linked Data visualization and exploration, data visualization in general as well as approaches for visualizing and interacting with particularly large datasets.

2.1 Linked Data Visualization and Exploration

Exploring and visualizing Linked Data is a problem that has been addressed by several projects. Dadzie et al. [1] analyzed some of those projects concluding that most of them are designed only for tech-users and do not provide overviews on the data.

Linked Data browsers such as *Disco* [6], *Tabulator* [7] or *Explorator* [8] allow users to navigate the graph structures and usually display property-value pairs in tables. They provide a view of a subject, or a set of subjects and their properties, but not any additional support getting a broader view of the dataset being explored. *Rhizomer* [9] provides an overview of the datasets and allows to interact with data through Information Architecture components such as navigation menus, breadcrumbs and facets. However, it needs to precompute some aggregated values and does not include visualizations.

Graph-based tools such as *Fenfire* [10], *RDF-Gravity*[1], *IsaViz*[2] provide node-link visualizations of the datasets and the relationships between them. Although this approach can help obtaining a better understanding of the data structure, in some cases graph visualization does not scale well to large datasets [11]. Sometimes the result is a complex graph difficult to manage and understand [12].

There are also *JavaScript-based tools* to visualize RDF. *Sgvizler*[3] renders the results of SPARQL queries into HTML visualizations such as charts, maps, treemaps, etc. However, it requires SPARQL knowledge in order to create RDF visualizations. *LODWheel* [13] also provides RDF visualizations for different data categories but it is focused on visualizing vocabularies used in DBpedia.

Other tools are restricted to visualizing and browsing concrete domains, e.g. *LinkedGeoData browser* [14] or *map4rdf*[4] for spatial data, *FoaF Explorer*[5] to visualize RDF documents adhering to the FOAF vocabulary.

To summarize, most of the existing tools make it difficult for non-technical users to explore linked data or they are restricted to concrete domains. None of them provide generic visualizations for RDF data. Consequently, it is still difficult for end users to obtain an overview of datasets, comprehend what kind of structures and resources are available and what properties resources typically have and how they are mostly related with each other.

[1] `http://semweb.salzburgresearch.at/apps/rdf-gravity`
[2] `http://www.w3.org/2001/11/IsaViz`
[3] `http://code.google.com/p/sgvizler/`
[4] `http://oegdev.dia.fi.upm.es/map4rdf/`
[5] `http://xml.mfd-consult.dk/foaf/explorer/`

2.2 Data Visualization

Regarding data visualization, most of the existing work is focused on categorizing visualization techniques and systems [15]. For example, [16] and [17] propose different taxonomies of visualization techniques. However, most visualization techniques are only compatible with concrete domains or data types [3].

Chi's *Data State Reference Model* [5] defines the visualization process in a generic way. It describes a process for transforming raw data into a concrete visualization by defining four data stages as well as a set of data transformations and operators. This framework provides a conceptual model that allows to identify all the components in the visualization process. In Section 3 we describe how this model serves as a starting point for our Linked Data Visualization Model.

2.3 Visualizing and interacting with large-scale datasets

For data analysis and information visualization, Shneiderman proposed a set of tasks based on the visual information seeking mantra: "*overview first, zoom and filter, then details on demand*" [18]. Gaining overview is one of the most important user tasks when browsing a dataset, since it helps end users to understand the overall data structure. It can also be used as starting point for navigation. However, overviews become difficult to achieve with large heterogeneous datasets, which is typical for Linked Data. A common approach to obtain an overview and support the exploration of large datasets is to structure them hierarchically [19]. Hierarchies allow users to visualize different abstractions of the underlying data at different levels of detail. Visual representations of hierarchies allow to create simplified versions of the data while still maintaining the general overview.

There are several techniques for visualizing hierarchical structures. One approach to provide high level overviews are *Treemaps* [20]. Treemaps use a rectangle to show the tree root and its children. Each child has a size proportional to the cumulative size of its descendants. They are a good method to display the size of each node in a hierarchy. In *CropCircles* [21] each node in the tree is represented by a circle. Every child circle is nested inside its parent circle. The diameter of the circles are proportional to the size of their descendants. *SpaceTrees* [22] combines the conventional layout of trees with zooming interaction. Branches of the tree are dynamically rescaled to fit the available space.

For datasets containing spatial data, another approach for obtaining an overview is to point resources on a map. Clustering [23] and organizing resources into hierarchies [24] are the most commonly used approaches for exploring large datasets on maps.

By applying these techniques it is possible to build interactive hierarchical visualizations that support the visual information seeking mantra proposed by Shneiderman.

3 Linked Data Visualization Model

In this section we present the Linked Data Visualization Model by first giving an overview, then formalizing key elements of the model and finally showcasing the model in action with a concrete example.

3.1 Overview

We use the *Data State Reference Model* (DSRM) proposed by Chi [5] as conceptual framework for our *Linked Data Visualization Model* (LDVM). While the DSRM describes the visualization process in a generic way, we instantiate and adopt this model with LDVM for the visualization of RDF and Linked Data. The names of the stages, transformations and operators have been adapted to the context of Linked Data and RDF. Figure 1 shows an overview of LDVM. It that can be seen as a pipeline, which originates in one end with raw data and results in the other end with the visualization.

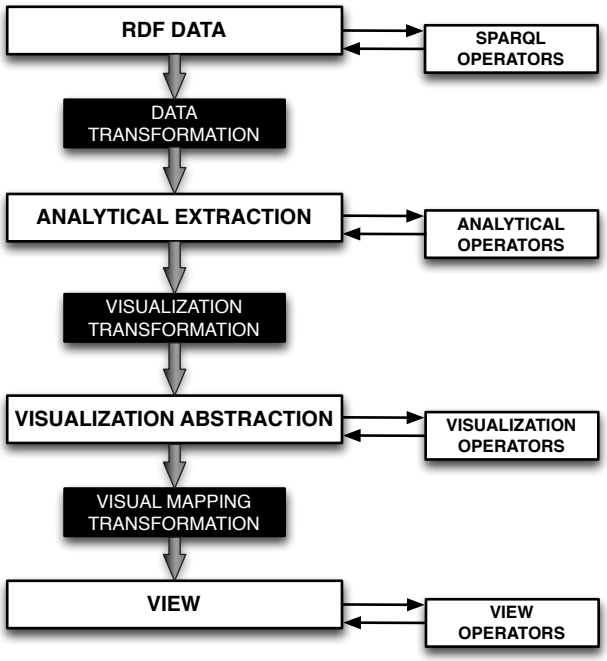

Fig. 1. High level overview of the Linked Data Visualization Model.

The LDVM pipeline is organized in four stages that data needs to pass through:

1. *RDF Data:* the raw data, which can be all kinds of information adhering to the RDF data model, e.g. instance data, taxonomies, vocabularies, ontologies, etc.
2. *Analytical extraction:* data extractions obtained from raw data, e.g. calculating aggregated values.
3. *Visual abstraction:* information that is visualizable on the screen using a visualization technique.
4. *View:* the result of the process presented to the user, e.g. plot, treemap, map, timeline, etc.

Data is propagated through the pipeline from one stage to another by applying three types of data transformation operators:

1. *Data transformation:* transforms raw data values into analytical extractions declaratively (using SPARQL query templates).
2. *Visualization transformation:* takes analytical extractions and transforms them into a visualization abstraction. The goal of this transformation is to condense the data into a displayable size and create a suitable data structure for particular visualizations.
3. *Visual mapping transformation:* processes the visualization abstractions in order to obtain a visual representation.

There are also operators within each stage that do not change the underlying data but allow to extract data from each stage or add new information. These are:

1. *SPARQL Operators:* SPARQL functions, e.g. SELECT, FILTER, COUNT, GROUP BY, etc.
2. *Analytical Operators:* functions applied to the data extractions, e.g. select a subset of data.
3. *Visualization Operators:* visual variables, e.g. visualization technique, sizes, colors, etc.
4. *View Operators:* actions applied to the view, e.g. rotate, scale, zoom, etc.

As shown in Figure 2, our model allows to connect different RDF datasets and different data extractions with different visualization techniques. Not all datasets are compatible with all data extractions, as well as each data extraction is only compatible with some visual configurations.

Each dataset has different data extraction possibilities, e.g. class hierarchy, property hierarchy, geospatial data, etc. Each data extraction can be visualized with different configurations, which contain information such as the visualization technique to use, colors, etc. Then, a concrete visualization is generated depending on the data extraction and the visual configuration.

In summary, the model is divided in two main blocks: data space and visual space. The RDF data stage, analytical extraction stage and data transformation belong to the data space, while visual abstraction stage, view stage and visual mapping transformation belong to the visual space. These two main blocks are connected by a visualization transformation.

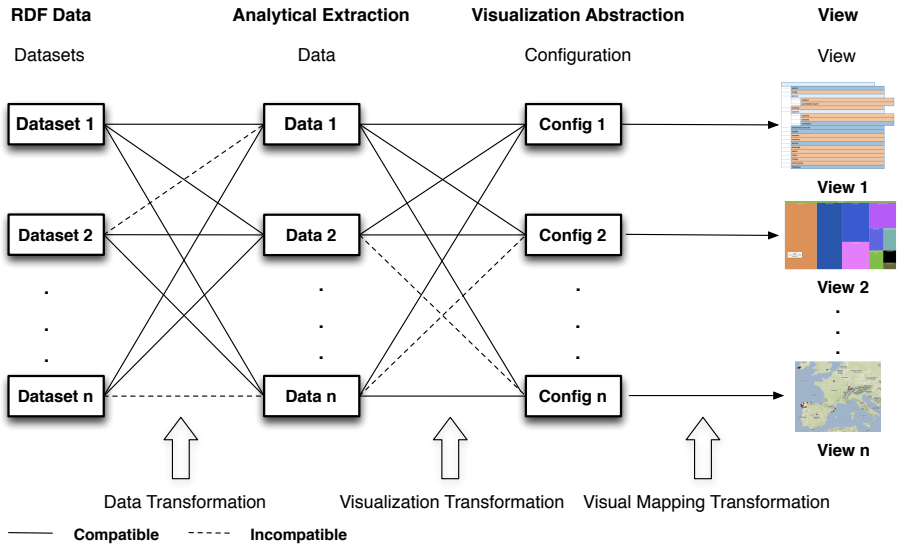

Fig. 2. Linked Data Visualization Model workflow.

3.2 Formalization

The RDF data model enables us to bind data to visualizations in a dynamic way while the LDVM allows to connect datasets with data extractions and visualizations. However, the compatibility between data and concrete visualizations needs to be determined.

In this section we formalize the extraction and generation of visualization data structures (in the Analytical Extraction Stage) as well as the visualization configuration (in the Visualization Abstraction Stage). Our visual data extraction structure is capturing the information required to (a) determine whether a certain dataset is compatible with a certain analytical extraction and (b) process a compatible dataset and extract a relevant subset (or aggregation) of the data as required by compatible visualization configurations. Our visual configuration structure determines wether a certain analytical extraction is compatible with a certain visualization technique.

Definition 1 (Visualization Data Extraction). *The Visualization Data Extraction defines how data is accessed for the purpose of visualizations. Our Visualization Data Extraction structure comprises the following elements:*

- *a compatibility check function (usually a SPARQL query), which determines whether a certain dataset contains relevant information for this data extraction,*
- *a possibly empty set D of data extraction parameters,*

– *a set tuples (q, σ_q), with q being a SPARQL query template containing placeholders mapped to the data extractions parameters D and σ_q being the signature of the SPARQL query result, i.e. the number of columns returned by queries generated from q and their types.*

Definition 2 (Visualization Configuration). *The Visualization Configuration is a data structure, which accommodates all configuration information required for a particular type of visualization. This includes:*

– *the visualization technique, e.g. Treemap, CropCircles, etc,*
– *visualization parameters, e.g. colors, aggregation,*
– *an aggregated data assembly structure, consisting of a set of relations r, each one equipped with a signature σ_r*

Definition 3 (Compatibility). *A Visualization Data Extraction vde is said to be compatible with a Visualization Configuration vc iff for each σ_r in vc there is a corresponding σ_q in vde, such that the types and their positions in σ_q and σ_r match.*

3.3 Example application of the model

In this section we present a concrete application of the Linked Data Visualization Model. It describes the process to create a Treemap visualization of the DBpedia [25] class hierarchy with its Data Stages, Transformations and Operators used in each Data Stage.

Data Stages

– *RDF Data:* the DBpedia dataset containing RDF triples and its ontology.
– *Analytical Extraction:* tuples containing the class URI, its number of instances and its children.
– *Visual Abstraction:* data structure containing the class hierarchy.
– *View:* a treemap representation of the class hierarchy.

Transformations

– *Data Transformation:* SPARQL queries to obtain the analytical extraction. The concrete SPARQL queries can be seen in the Example 1.
– *Visual Transformation:* a function that creates the class hierarchy from the tuples obtained in the Analytical Extraction Stage.
– *Visual Mapping Transformation:* a function that transforms the class hierarchy into a Treemap using a concrete visualization library.

Stage Operators

- *SPARQL Operators:* DISTINCT, COUNT, GROUP BY, AS, FILTER, OPTIONAL.
- *Analytical Extraction Operators:* functions to parse the SPARQL queries results and convert them to a concrete format and structure.
- *Visual Abstraction Operators:* size of the treemap, colors.
- *View:* zoom in, zoom out.

Example 1 (Visualization Data Extraction).

Compatibility check function: the following SPARQL query is executed in order to check if this data extraction is compatible with the dataset. If the query returns a result it means that this information is present in the dataset and the data extraction is compatible with it.

```
ASK
WHERE {
  ?c  rdf:type  ?class .
  FILTER (?class=owl:Class || ?class=rdfs:Class)
}
LIMIT 1
```

SPARQL Query to obtain root elements: the following SPARQL query obtains the root elements of the class hierarchy and their labels if they exist.

```
SELECT DISTINCT ?root ?label
WHERE {
  ?root  rdf:type  ?class .
    FILTER (?class=owl:Class || ?class=rdfs:Class) .
  OPTIONAL {?root  rdfs:subClassOf  ?super .
    FILTER (?root!=?super && ?super!=owl:Thing && ?super!=rdfs:
        Resource && !isBlank(?super))
  } .
  OPTIONAL {
    ?root  rdfs:label  ?label  .
    FILTER(LANG(?label)='en' || LANG(?label)='')
  } .
  FILTER (!bound(?super) && isURI(?root) && !isBlank(?root) && ?root
      !=owl:Thing )
}
```

SPARQL Query to obtain children per element: the following SPARQL query is used to obtain the parent of each class in the hierarchy and their labels if they exist.

```
SELECT DISTINCT ?sub ?label ?parent
WHERE {
  ?sub  rdf:type  ?class .
    FILTER (?class=owl:Class || ?class=rdfs:Class) .
  ?sub rdfs:subClassOf ?parent .
    FILTER(!isBlank(?parent) && ?parent!=owl:Thing) .
  OPTIONAL {
    ?sub  rdfs:subClassOf  ?sub2 .
    ?sub2  rdfs:subClassOf  ?parent .
    OPTIONAL {
      ?parent  owl:equivalentClass  ?sub3 .
      ?sub3  rdfs:subClassOf  ?sub2
    } .
    FILTER (?sub!=?sub2 && ?sub2!=?parent && !isBlank(?sub2) && !
        bound(?sub3) )
```

```
  } .
  OPTIONAL {
    ?sub  rdfs:label  ?label .
    FILTER(LANG(?label)='en')
  } .
  FILTER (!isBlank(?sub) && !bound(?sub2))
}
```

SPARQL Query to obtain number of resources per element: the following SPARQL query obtains the number of resources of each class in the hierarchy.

```
SELECT ?c (COUNT(?x) AS ?n)
WHERE {
  ?x  rdf:type  ?c .
  ?c  rdf:type  ?class .
  FILTER (?class=owl:Class || ?class=rdfs:Class)
}
GROUP BY ?c
```

SPARQL Query to obtain a list of resources per element: the following SPARQL query obtains a list of resources that belong to a concrete class (in placeholder %s)

```
SELECT DISTINCT ?x ?label
WHERE {
  ?x  rdf:type  %s .
  OPTIONAL {
    ?x  rdfs:label  ?label .
    FILTER(LANG(?label)='en' || LANG(?label)='')
  }
}
LIMIT 10
```

Example 2 (Visualization Configuration). The visualization Configuration structure contains:

- Visualization Technique: Treemap,
- Visualization parameters:
  - Colors: random,
  - Aggregation[6] : yes.

4 Implementation

Based on the Linked Data Visualization Model proposed, we have implemented a prototype called LODVisualization[7]. It allows to explore and interact with the Data Web through different visualizations. The visualizations implemented suport the *Overview* task proposed by Shneiderman [18] in his visual information seeking mantra. This way, our prototype serves not only as a proof of concept of our LDVM but also provides useful visualizations of RDF. These visualizations allow users to obtain an overview of RDF datasets and realize what the data is about: their main types, properties, etc.

[6] group those classes that have an area too small to show in the Treemap

[7] `http://lodvisualization.appspot.com/`

LODVisualization is developed using Google App Engine[8] (GAE). Google App Engine is a cloud computing platform for developing and hosting web applications in Google's infrastructure. Figure 3 shows the architecture of LODVisualization.

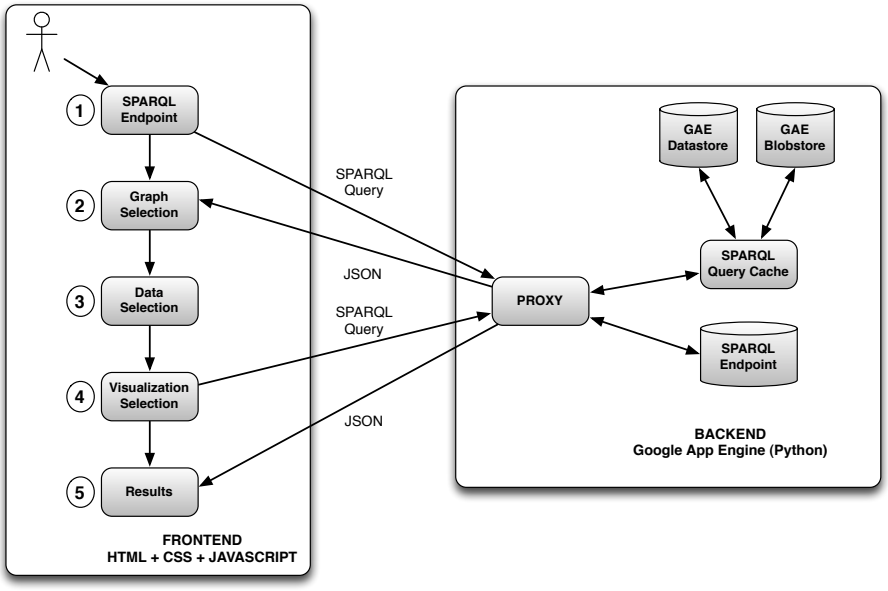

Fig. 3. LODVisualiation architecture

Steps 1 and 2 correspond to the RDF Data Stage, in which users can enter an SPARQL endpoint and select those graphs to visualize. Step 3 corresponds to the Data Extraction Stage, in which users can choose a data extraction available for the graphs selected. Step 4 corresponds to the Visualization Abstraction Stage, in which users can select a visualization technique compatible with the data extraction and some visualization parameters such as colors, size, etc. Finally, step 5 corresponds to the View Stage and presents a concrete visualization to the users.

The frontend is developed using HTML, CSS and Javascript, while the backend is developed in Python. Most visualizations are created using JavaScript Infovis Toolkit[9] and D3.js [26]. We have chosen them because they allow to create rich and interactive JavaScript visualizations and they include also a free license. We also use Google Maps and OpenStreetMap to create Geospatial Visualizations. Data is transferred between SPARQL endpoints, the server and the

[8] `https://developers.google.com/appengine/`
[9] `http://thejit.org/`

client using JSON, which is the data format used by those libraries. Most of SPARQL endpoints support JSON and can convert RDF to it, making it easy to integrate it into Javascript.

Instead of querying directly SPARQL endpoints using AJAX, queries are generated in the client side but sent to the server. The server acts as proxy in order to avoid AJAX Cross-domain requests. At the same time, the server provides a cache memory implemented using GAE Datastore and GAE Blobstore. For each SPARQL query we generate a MD5 hash and we store it with the query results. The Blobstore is used to save those results whose size exceeds 1Mb because the Datastore has this limitation. When the server receives a query, the results are obtained from cache if it is already there. Otherwise, the query is sent to the SPARQL endpoint using the GAE URL Fetch API. Then, the hash of the query and its results are stored in cache. A cache memory is useful to prevent executing the same queries when the users want to visualize the same data extraction with different visualization techniques.

Our implementation includes so far 4 different data extraction types and 7 different visualization techniques. The data extraction possibilities are: class hierarchy, property hierarchy, SKOS concepts hierarchy and geospatial data. The visualization techniques implemented are: Treemap, Spacetree, CropCircles, Indented Tree Google Maps and OpenStreetMap. Figure 4 shows a concrete visualization example with the following parameters:

- Dataset: DBpedia.
- Data Extraction: class hierarchy.
- Visualization Configuration: Treemap, aggregation, random colors.

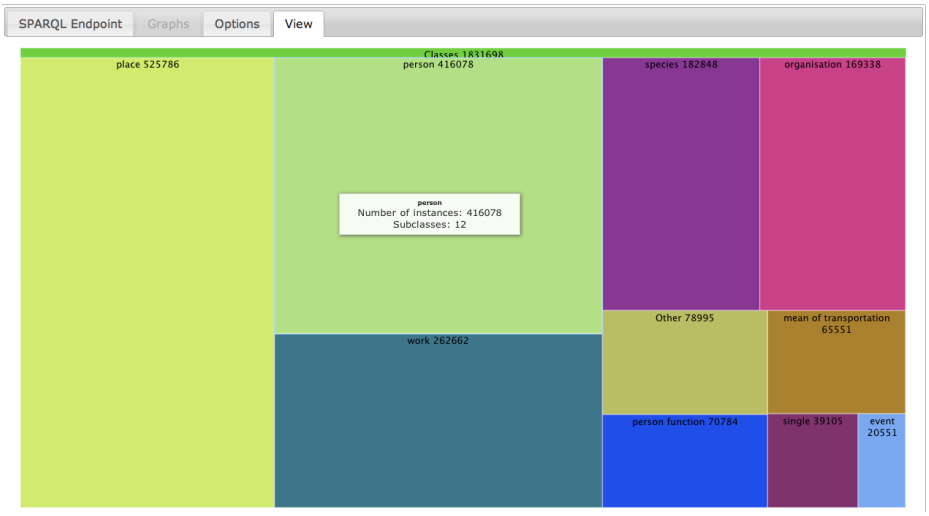

Fig. 4. DBpedia class hierarchy Treemap

LODVisualization is compatible with most of SPARQL endpoints as long as they support JSON and SPARQL 1.1[10]. Most of the data extraction queries use aggregate functions such as `COUNT` or `GROUP BY`, which are implemented from this version. Our implementation has a limitation: the response of each SPARQL query must be in less than 60 seconds. This is the maximum deadline for requests using the GAE URL Fetch API. Nevertheless, longer requests could be performed using the GAE Task Queue API.

5 Evaluation

We have evaluated our implementation of the Linked Data Visualization Model with different datasets, data extractions and visualizations. The evaluation was performed with Google Chrome version 19.0.1084.46. Table 1 shows a summary of the evaluation results.

ID	Dataset	Data Extraction	Visualization Configuration	Execution time (s)	
				w/o cache	w/ cache
1	DBpedia	Class hierarchy	Treemap	3.58	0.87
2	DBpedia	Class hierarchy	Crop Circles	3.35	0.76
3	DBpedia	SKOS concept hierarchy	Treemap	48.79	16.96
4	DBpedia	Spatial data	Google Maps	2.05	0.37
5	Dbpedia	Spatial data	OpenStreetMaps	2.03	0.79
6	Wine Ontology	Property hierarchy	Indented Tree	1.95	0.57
7	Wine Ontology	Property hierarchy	Space Tree	2.45	0.70
8	LinkedMDB	Class hierarchy	Treemap	3.28	0.61
9	WWW 2011	Class hierarchy	Crop Circles	2.53	0.75
10	AGRIS – FAO	SKOS concept hierarchy	Treemap	28.72	8.54

Table 1. Evaluation results summary

Each concrete visualization was executed 10 times without cache, 10 times with cache and we calculated the average time of the executions. As it can be seen, most of the visualizations can be generated in real time and the use of a cache memory reduces the execution time. However, in experiments #3 and #10 (SKOS concept hierarchies) the results had to be stored in the GAE Blobstore instead of the GAE Datastore due to their size. Accessing the Blobstore to retrieve those results is much slower than accessing the Datastore.

Table 2 shows the timings for each visualization divided into the three transformations proposed in the LDVM: data transformation, visual transformation and visual mapping transformation.

[10] `http://www.w3.org/TR/sparql11-query/`

ID	Data Transformation - SPARQL Query Templates (s)				Visual Transformation (s)	Visual Mapping Transformation (s)
	#1	#2	#3	Total		
1	1.4336	1.2741	0.7305	3.4382	0.0190	0.1228
2	1.2194	1.3423	0.7123	3.2740	0.0140	0.0620
3	3.5202	5.5424	35.9858	45.0484	0.4588	3.2900
4	2.0028	-	-	2.0028	0.0148	0.0356
5	1.6008	-	-	1.6008	0.0100	0.4266
6	0.7792	0.4860	0.6712	1.9364	0.0022	0.0130
7	0.98875	0.6757	0.5650	2.2295	0.0015	0.0372
8	0.9554	0.7334	1.5442	3.233	0.0086	0.0482
9	0.8914	0.8924	0.7016	2.4854	0.0102	0.0430
10	6.2942	6.4924	14.1782	26.9648	0.5428	1.2184

Table 2. Timing for each transformation

Creating different visualizations for the same data extraction takes a similar time. This is because most of the execution time corresponds to the data transformation. The visual transformation timing depends on the size of the results to process. In the same way, the visual mapping transformation depends on the number of items to visualize. This is particularly high in experiments #3 and #10, with a huge hierarchy of SKOS concepts.

However, it is important to highlight that the execution times are not really relevant for this evaluation. They depend mainly on the complexity of the SPARQL queries required for the data extraction as well as on the availability of SPARQL endpoints or Google App Engine servers. The goal of our evaluation was to prove that the LDVM can be applied to different datasets providing different data visualizations. All these visualization examples are available in the website and it easy to create new ones.

6 Conclusions and Future Work

We have presented the Linked Data Visualization Model (LDVM) that can be applied to create visualizations of RDF data. It allows to connect different datasets, different data extractions and different visualizations in a dynamic way. Applying this model, developers and designers can get a better understanding of the visualization process with data stages, transformations and operators. This model can offer user guidance on how to create visualizations for RDF data.

We have implemented the model into an application where it is possible to create generic visualizations for RDF. In our implementation we provide visualizations that support the overview task proposed by Shneiderman in his visual seeking mantra. These visualizations are useful for getting a broad view of the datasets, their main types, properties and the relationships between them.

Our future work focuses on complementing the Linked Data Visualization Model with an ontology. A visualization ontology can help during the matching process between data and visualizations, capture the intermediate data structures that can be generated and choose the visualizations more suitable for each data structure.

Regarding our implementation of the model, we plan to develop new data extractions and visualizations for them, e.g. charts or timelines. Having more data extractions and visualizations will prove that the LDVM can be widely applied.

Finally, based on the LDVM we want to develop a dashboard for visualizing and interacting with linked data through different visualizations and other components. However, other Information Architecture components [27] such as facets or breadcrumbs are necessary to support the other tasks for data analysis and exploration.

References

1. Aba-Sah Dadzie and Matthew Rowe. Approaches to visualising linked data: A survey. *Semantic Web*, 2(2):89–124, 2011.
2. Vladimir Geroimenko and Chaomei Chen, editors. *Visualizing the Semantic Web*. Springer, 2002.
3. Stuart K. Card, J. D. Mackinlay, and Ben Shneiderman. *Readings in Information Visualization: Using Vision to Think*. Academic Press, London, 1999.
4. Jeffrey Heer, Michael Bostock, and Vadim Ogievetsky. A tour through the visualization zoo. *Queue*, 8(5):20:20–20:30.
5. Ed H. Chi. A taxonomy of visualization techniques using the data state reference model. In *Proceedings of the IEEE Symposium on Information Vizualization 2000*, INFOVIS '00, pages 69–, Washington, DC, USA, 2000. IEEE Computer Society.
6. Uldis Bojars, Alexandre Passant, Frederick Giasson, and John Breslin. An architecture to discover and query decentralized RDF data. volume 248 of *CEUR Workshop Proceedings ISSN 1613-0073*, June 2007.
7. Tim Berners-lee, Yuhsin Chen, Lydia Chilton, Dan Connolly, Ruth Dhanaraj, James Hollenbach, Adam Lerer, and David Sheets. Tabulator: Exploring and analyzing linked data on the semantic web. In *In Proceedings of the 3rd International Semantic Web User Interaction Workshop*, 2006.
8. Experimenting with explorator: a direct manipulation generic rdf browser and querying tool. In *Workshop on Visual Interfaces to the Social and the Semantic Web (VISSW2009)*, February 2009.
9. J.M. Brunetti, R. Gil, and R. Garcia. Facets and pivoting for flexible and usable linked data exploration. Crete, Greece, May 2012.
10. Tuukka Hastrup, Richard Cyganiak, and Uldis Bojars. Browsing linked data with fenfire, 2008.
11. Adapting Graph Visualization, Flavius Frasincar, Ru Telea, and Geert jan Houben. Adapting graph visualization techniques for the visualization of rdf data. In *of RDF data, Visualizing the Semantic Web, 2006*, pages 154–171, 2005.
12. David Karger and MC Schraefel. The pathetic fallacy of RDF. Position Paper for SWUI06, 2006.

13. Magnus Stuhr, Roman Dumitru, and David Norheim. *LODWheel - JavaScript-based Visualization of RDF Data* -, pages 1–12. 2011.
14. Claus Stadler, Jens Lehmann, Konrad Höffner, and Sören Auer. Linkedgeodata: A core for a web of spatial open data. *Semantic Web Journal*, 2011.
15. Maria Cristina Ferreira de Oliveira and Haim Levkowitz. From visual data exploration to visual data mining: A survey. *IEEE Transactions on Visualization and Computer Graphics*, 9:378–394, 2003.
16. S. K. Card and J. Mackinlay. The structure of the information visualization design space. In *Proceedings of the 1997 IEEE Symposium on Information Visualization (InfoVis '97)*, INFOVIS '97, pages 92–, Washington, DC, USA, 1997. IEEE Computer Society.
17. E. H. H. Chi, P. Barry, J. Riedl, and J. Konstan. A spreadsheet approach to information visualization. In *Proceedings of the 1997 IEEE Symposium on Information Visualization (InfoVis '97)*, INFOVIS '97, pages 17–, Washington, DC, USA, 1997. IEEE Computer Society.
18. Ben Shneiderman. The eyes have it: A task by data type taxonomy for information visualizations. In *IEEE Visual Languages*, number UMCP-CSD CS-TR-3665, pages 336–343, College Park, Maryland 20742, U.S.A., 1996.
19. Niklas Elmqvist and Jean-Daniel Fekete. Hierarchical aggregation for information visualization: Overview, techniques, and design guidelines. *IEEE Trans. Vis. Comput. Graph.*, 16(3):439–454, 2010.
20. Ben Shneiderman. Tree visualization with tree-maps: 2-d space-filling approach. *ACM Trans. Graph.*, 11(1):92–99, January 1992.
21. Taowei David Wang and Bijan Parsia. Cropcircles: Topology sensitive visualization of owl class hierarchies. In *In Proc. of the 5th International Conference on Semantic Web*, pages 695–708, 2006.
22. Catherine Plaisant, Jesse Grosjean, and Benjamin B. Bederson. Spacetree: Supporting exploration in large node link tree, design evolution and empirical evaluation. In *Proceedings of the IEEE Symposium on Information Visualization (InfoVis'02)*, INFOVIS '02, pages 57–, Washington, DC, USA, 2002. IEEE Computer Society.
23. Natalia Andrienko. Interactive visual clustering of large collections of trajectories. In *Working Notes of the LWA 2011 - Learning, Knowledge, Adaptation*, 2011.
24. Heidrun Schumann and Christian Tominski. Analytical, visual and interactive concepts for geo-visual analytics. *J. Vis. Lang. Comput.*, 22(4):257–267, August 2011.
25. Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, and Zachary Ives. Dbpedia: A nucleus for a web of open data. In *In 6th Int?l Semantic Web Conference, Busan, Korea*, pages 11–15. Springer, 2007.
26. Michael Bostock, Vadim Ogievetsky, and Jeffrey Heer. D3: Data-driven documents. *IEEE Trans. Visualization & Comp. Graphics (Proc. InfoVis)*, 2011.
27. Josep Maria Brunetti and Roberto García. Information architecture automatization for the semantic web. In *Proceedings of the 13th IFIP TC 13 international conference on Human-computer interaction - Volume Part IV*, INTERACT'11, pages 410–413, Berlin, Heidelberg, 2011. Springer-Verlag.