Visualizing Semantic Data

Michael Luggen

Synonyms

- Visualizing Linked Data
- Visualizing RDF Data
- Automated creation of Infographics

Definition

Visualizing Semantic Data describes the task of using the additional self describing features of semantic data (Linked Data / RDF) to inform the process of creating vector or bitmap drawings.

Introduction

Visualizations are representations of data solely created for humans. The primary objective of a visualization is to clearly and efficiently communicate information to a target audience. A visualization allows us to focus on particular aspects of data by choosing specific types of primitives and aggregates. Furthermore a visualization optimizes for efficiently delivering the intended message to the end-user.

Creating meaningful and efficient visualizations of data is a difficult task. It integrates the understanding of the data at hand, the knowledge of valid aggregations and transformations and finally the selection of the right type of visualization to convey the intended message. There exists multiple professions like infographic design, data journalism and data science specializing in these tasks. These professions rely on a variety of tools to create visualizations. The tools are informed by a plethora of implicit knowledge, based on raw data mostly.

With semantic data, i.e. data which is provided with additional information regarding its form and content, new possibilities open up for data visualization. The modeling of the data and its rela-

tions are defined by well known vocabularies and ontologies. Furthermore, with the use of RDF (see https://www.w3.org/RDF/), semantic data can be represented in a common representation regardless of the form of its original or underlying data model.

Examples of such data models include:

- Dimensionally organized data (linear data models): Time-series, Statistical data, Geographic data
- Structural data: trees, taxonomies and graph data
- Annotated Multimedia: Images, Audio, Video, 3d Models

A tool without any external information and that is also missing knowledge on the data model at hand cannot propose any sensible visualization. Thus, there exist several tools for creating visualizations specific to a data model.

The process adopted to create a visual representation based on semantic data can be described end-to-end; this provides the inherent advantage that the outcome is reproducible by the system without further external intervention by the user or developer in case of corrections or updates to the underlying data.

Future research directions could propose systems which can fulfill the demand of efficient and effective visualizations by allowing users to craft visual representations based on their formulated intent combined with the inspection of the provided data.

Because semantic data in RDF is represented as graph data, a number of methods were proposed for visualizing this graph data as graph drawings (nodes-link diagrams) (Graziosi et al 2017). Recent examples on the web are VOWL (http://vowl.visualdataweb.org/) and LodLive (http://en.lodlive.it/). However, researchers have argued that there exist only a small number of specific use cases (e.g. exploration of ontologies (Lohmann et al 2016)) where the graph representation is actually efficient for the task at hand. For most tasks, other forms of visualizations tend to be more efficient (Dadzie and Pietriga 2016).

In the following sections, the current state of research and available tools for the creation of visualizations with semantic data (In our example represented in RDF) are described. First, a number of *requirements regarding the data model* of the semantic data are introduced. Then, a *taxonomy for the overall task* of creating visualizations based on semantic data is provided. In addition, the different *steps necessary to create visualizations* based on semantic data are discussed. Finally, an overview on the *spectrum of tools* used for visualizations, from hand-crafted to automated approaches, is provided.

Semantic Data Requirements

Semantic data provides information to machines which help to understand the data itself. Apart from having the data expressed in a valid form like RDF, it is necessary for the data to be modeled by an ontology or described in an RDF Schema. This allows the reuse of any visualization on similar datasets following the same ontology. In the following, a selection of ontologies which are leveraged for visualization techniques are described. The ontologies are organized by

the data model they provide for visualization.

Dimensional

Dimensional data organizes related data points on a dimension; in the case of semantic data, it additionally describes its content. Typical examples of dimensional are the *time* dimension or dimensions from a data-specific categorical attribute like *gender* or *age*.

A good starting point for arbitrary dimensional data is the *Data Cube Vocabulary* (https://www.w3.org/TR/vocab-data-cube/). Not only does this vocabulary allow for a clear definition of the dimensions and properties inside a dataset, it also defines additional descriptions like titles, dimension descriptions, and units which can be attached to the data points, dimensions and datasets respectively.

Another base for dimensional data is to use the standard mode output of *csv2rdf* (https://www.w3.org/TR/csv2rdf/). This is especially useful as the data to visualize is often provided in CSV format anyway.

Hierarchical / Structural

For the modeling of hierarchical and structural information, the use of the *SKOS Ontology* (https://www.w3.org/TR/skos-reference/) is widely spread. It is also feasible to represent structural information directly with the *OWL Web Ontology Language* (https://www.w3.org/TR/owl-overview/).

Geographical

Geo-spatially located data points often use the simple *Basic Geo Ontology* (https://www.w3.org/2003/01/geo/). For more complex geospatial data, one can refer to the *GeoSPARQL* (http://www.opengeospatial.org/standards/geosparql) documentation which is using *Well Known Text (WKT)* (http://www.geoapi.org/3.0/javadoc/org/opengis/referencing/doc-files/WKT.html)

Miscellaneous

Further media data (pictures, video, or audio) can be described and annotated with the *Web Annotation Vocabulary* (https://www.w3.org/TR/annotation-vocab/). The power of semantic data also comes with the possibility to combine and mix different data models. This is possible as all the data models introduce above are organized by the same RDF formalisms.

Taxonomy

In this section, a high level taxonomy with the goal to situate the different available approaches is provided. The taxonomy is influenced by prior work on visualization taxonomies: Brunetti et al

(2013) provide a formal framework in this context while Graziosi et al (2017) focus on the visualization of graph drawings.

The task of creating visualizations based on semantic data is split into the following steps.

1. View Selection: When necessary, a specific subset of the available data is chosen.
2. Visual Primitive Selection: A fitting type of visualization is selected.
3. Data Mapping: Data then needs to be projected onto the Visual Primitive.

At the end of this section interactive visualizations in line with this taxonomy are shortly discussed.

1. Selection of the Data View

The semantic data to be visualized can be accessed in multiple ways, from data dump files through server calls or a generic query language.

Typically, the data to be visualized is a part of all the available data points in a dataset. Hence, there is the need to select a view on the data, which is a subset of all available data. The decision on which data points are part of the visualization is determined by the application logic which also handles the interaction with the visualization itself.

The selection of the Data View is dependent of primitives available for data access and filtering. In case the semantic data is preprocessed in a file, the selection was probably already done during its creation process. In case there is a (REST Style) API defined to provided the semantic data, the possible Views are then defined through the API design.

On the contrary, if semantic data is consumed through a query language (like SPARQL), the application logic can define the view on the data in more detail. The generated visualizations also might have the potential to interact with the view selection mechanism in order to get updated iteratively.

2. Selection of the Visual Primitive

Once the decision is made about which data points should be visualized, selecting a suitable visual primitive is necessary. A visual primitive is a description of how to transform (render) the structured data on a canvas to create a visualization. Two approaches to select the appropriate visual primitives can be identified: an explicit selection during the data creation process, or an implicit selection at run time.

Generally, the selection of the visual primitive is *explicitly* defined, either in the programming code itself, or through annotating the data with hints which visualization to use.

For systems where the selection is *implicit*, the selection of the visual primitive becomes a ranking problem based on a catalog of available visual primitives. Rule based systems leverage properties of the semantic data to decide on a visual primitive. Many properties can be leveraged in that sense, including the size or the dimension of data objects, or the level of a hierarchy. In addition, the rule engine can also leverage features from the semantic model itself, for example the use of specific ontologies (as introduced in the section above).

Rule-based engines can be further informed by the context of the application or use-cases, including the mode of interaction with the user (role, language) and the current device(-type) she uses (desktop, mobile).

Future research is needed to provide applications to solve the selection of visual primitives with machine learning approaches based on the features described above.

Explicit and implicit approaches are not mutually exclusive and systems profit from a combination of both these methods.

3. Data Mapping

Once the view on the data and the visual primitive have been fixed, both need to be connected.

Projection of the Data

For the majority of visual primitives (e.g. line and bar charts), the data needs to be projected from the graph representation of the Semantic Data onto one or multiple continuous dimensions.

If the data is consumed through SPARQL, this step is often solved early on by using a SELECT query. The SELECT query provides by design as an output a projection of the data to a table structure. This provides a dimensional structure which can be consumed by the mentioned visual primitives. The disadvantage of using a SELECT query is that the output of the query is no longer Semantic Data but rather a flat table. This has the drawback that additional annotations on the data that might be useful for the visualization are stripped away.

This is why several tools making an implicit visual selection use CONSTRUCT queries in SPARQL instead. The output of such a query is a subset of the source of the Semantic Data it but keeps all semantic aspects.

Mapping based on Semantic Data

If the decision is made to keep the Semantic Data intact all the way to the creation of the Visual Primitive, the need to bind the data from the graph structure to the input format of a visual primitive arises. Two approaches are possible in this context.

The more traditional approach is to leverage the information which is held in a TBOX (i.e., the data model, also called an ontology or a schema) to create the mapping for the visual primitive. Thus, the explicit conceptual model of a data structure which informs the mapping is kept. The advantage is that the ontology at hand defines the form of the content itself. This approach has the disadvantage that explicit formulation of the ontology needs to be available during the mapping process.

There are also frameworks which simply rely on the ABOX (the instance data itself). Here, the mapper inspects the data and the input expected by the visual primitive to discover a suitable mapping. In this case the expected input of the visual primitive figures as an implicit TBOX.

Interaction

The interaction with the visual primitives is generally orchestrated through the application logic. In addition, the different visual primitives often come with elements to interact with the visualization itself. Possible interaction elements are filters on dimensions and values, controls for the selection of dimensions, or controls on the extent to which the data should be shown in the visual primitive.

These controls can either adapt the output of the visual primitive, or trigger the demand for another visual primitive altogether.

Tools

In this section, a selection of the available research and tools related to the creation of visualizations based on Semantic Data is provided, with a focus on tools that create graphics rendered in web browsers. To set a common base for all tools, an overview of the facilities from a web browser to enable drawing of visual elements is first given.

In the first two sections below (*Visualization Libraries and Frameworks* and *SPARQL specific frameworks*), approaches where the visualization is explicitly defined are discussed. In the third section (*Selection and Mapping Frameworks*) the current state of research for utilizing implicit selection of visualization primitives is discussed.

Table 1 provides an overview of current tools and sets them in relation with the taxonomy introduced above.

The *HTML5 Canvas Element*, the *HTML WebGL Element* and the *HTML SVG Element* are the most basic elements for creating visualizations. While the HTML5 Canvas is a pixel-based canvas which provides basic drawing facilities, WebGL provides a 3D context based on OpenGL to draw graphics, while SVG allows to create vector-based visualizations by dynamically creating SVG files. Web-based libraries and frameworks use one or multiple of these elements to perform the visualization inside a web browser.

Visualization Libraries and Frameworks

Visualization libraries differ greatly in their flexibility. A plethora of libraries (`https://developers.google.com/chart/`, `http://www.chartjs.org/`, `https://www.highcharts.com/`) provide catalogs of visual primitives, which expect the input data to be in a specific format. Most of these libraries provide only basic visualization possibilities.

The most prominent example of a more flexible framework is *D3.js* (`http://d3js.org`). *D3.js* provides developers with a set of tools to create visual primitives that are highly customizable.

SPARQL specific frameworks

Sgvizler Skjaeveland (2012) is the first widely used framework to create visualizations directly based on Semantic Data. Sgvizler fetches data from a SPARQL endpoint with SELECT queries and relies heavily on Google

Table 1 Presented Tools located in the introduced taxonomy.

Tool/Framework	View Selection	Visual Primitive	Mapping
Sgvizler	SELECT	explicit	hard coded catalog
D3SPARQL	SELECT	explicit	hard coded catalog
d3-sparql	SELECT	n/a	n/a
Tabulator	none (Browser)	implicit / rule-based	in code
Balloon Synopsis	Ontology Template	implicit / rule-based	Handlebars
Uduvudu	CONSTRUCT	implicit / rule-based	Matchers
RSLT	SQM/DQM	implicit / rule-based	HTML Templates
Linked Data Reactor	Scopes	implicit / rule-based	Web Component
Phuzzy	none (Browser)	implicit / rule-based	in code (plugin)

Charts for the visual primitives. It also provides an interface to write custom plug-ins which to create arbitrary visualizations.

The *D3SPARQL* project Katayama (2014) is comparable to Sgvizler in its functionality, as it provides a set of Visual Primitives based on *D3.js*.

Finally, the D3.js component *d3-sparql* (`https://github.com/zazuko/d3-sparql`) seamlessly integrates in the D3.js code base. It is a drop-in replacement for the d3-csv component. In that way, existing D3.js visualization can be informed through data originating from a SPARQL endpoint.

Selection and Mapping Frameworks

The frameworks presented in the following all provide an implicit selection of the visual primitive.

Tabulator Berners-Lee et al (2006) introduced a generic data browser, along with the concept of an implicit selection of a user interface component informed through Semantic Data. Some of these components were featuring visual primitives.

Balloon Synopsis (Schlegel et al 2014) is a jQuery Plugin which allows to easily enhance a website with semantic data. For the mapping *Ontology Templates* were introduced.

In Uduvudu (Luggen et al 2015) a framework for adaptive user interface generation is proposed. The Uduvudu framework uses a two-step ruled-based approach to select the user interface elements and the visual primitives. By separating the selection and the rendering step, it puts a focus on the re-usability by maximizing dynamic composition capabilities. Furthermore, the framework can take the context of the end-user (user preferences, used device, language) into account whenever available.

RSLT (Peroni and Vitali 2015) provides a library inspired by the XSLT transformation language. The semantic data is transformed from its RDF model by using templates. The templates are matched with an newly proposed selector language.

Linked Data Reactor (Khalili 2016) is a framework to build reactive applications realized with hierarchical *Web Components*. The mapping to a compo-

nent can be done on the level of a dataset, resource, property or value.

Phuzzy (Regalia et al 2017) introduces the capability of dynamically downloading plugins providing rendering facilities for visual primitives.

References

Berners-Lee T, Chen Y, Chilton L, Connolly D, Dhanaraj R, Hollenbach J, Lerer A, Sheets D (2006) Tabulator: Exploring and analyzing linked data on the semantic web. In: Proceedings of the 3rd international semantic web user interaction workshop, Citeseer, vol 2006, p 159

Brunetti JM, Auer S, García R, Klímek J, Nečaský M (2013) Formal Linked Data Visualization Model. In: Proceedings of International Conference on Information Integration and Web-based Applications & Services, ACM, New York, NY, USA, IIWAS '13, pp 309:309–309:318, DOI 10.1145/2539150.2539162, URL http://doi.acm.org/10.1145/2539150.2539162

Dadzie AS, Pietriga E (2016) Visualisation of Linked Data – Reprise | www.semantic-web-journal.net. URL http://www.semantic-web-journal.net/content/visualisation-linked-data-%E2%80%93-reprise

Graziosi A, Iorio AD, Poggi F, Peroni S (2017) Customised Visualisations of Linked Open Data. In: Ivanova V, Lambrix P, Lohmann S, Pesquita C (eds) Proceedings of the Third International Workshop on Visualization and Interaction for Ontologies and Linked Data co-located with the 16th International Semantic Web Conference (ISWC 2017), Vienna, Austria, October 22, 2017, CEUR-WS.org, CEUR Workshop Proceedings, vol 1947, pp 20–33, URL http://ceur-ws.org/Vol-1947/paper03.pdf

Katayama T (2014) D3sparql: JavaScript Library for Visualization of SPARQL Results. In: SWAT4LS, Citeseer

Khalili A (2016) Linked Data Reactor: a Framework for Building Reactive Linked Data Applications. In: Troncy R, Verborgh R, Nixon LJB, Kurz T, Schlegel K, Sande MV (eds) Joint Proceedings of the 4th International Workshop on Linked Media and the 3rd Developers Hackshop co-located with the 13th Extended Semantic Web Conference ESWC 2016, Heraklion, Crete, Greece, May 30, 2016, CEUR-WS.org, CEUR Workshop Proceedings, vol 1615, URL http://ceur-ws.org/Vol-1615/semdevPaper4.pdf

Lohmann S, Negru S, Haag F, Ertl T (2016) Visualizing ontologies with VOWL. Semantic Web 7(4):399–419, DOI 10.3233/SW-150200, URL https://content.iospress.com/articles/semantic-web/sw200

Luggen M, Gschwend A, Anrig B, Cudré-Mauroux P (2015) Uduvudu: a Graph-Aware and Adaptive UI Engine for Linked Data. In: Bizer C, Auer S, Berners-Lee T, Heath T (eds) Proceedings of the Workshop on Linked Data on the Web, LDOW 2015, co-located with the 24th International World Wide Web Conference (WWW 2015), Florence, Italy, May 19th, 2015, CEUR-WS.org, CEUR Workshop Proceedings, vol 1409, URL http://ceur-ws.org/Vol-1409/paper-07.pdf

Peroni S, Vitali F (2015) RSLT: R. D. F. Stylesheet Language Transformations. In: Verborgh R, Sande MV (eds) Proceedings of the ESWC Developers Workshop 2015 co-located with the 12th Extended Semantic Web Conference (ESWC 2015), Portorož, Slovenia, May 31, 2015, CEUR-WS.org, CEUR Workshop Proceedings, vol 1361, pp 7–13, URL http://ceur-ws.org/Vol-1361/paper2.pdf

Regalia B, Janowicz K, Mai G (2017) Phuzzy.link: A SPARQL-powered Client-Sided Extensible Semantic Web Browser. In: Proceedings of the Third International Workshop on Visualization and Interaction for Ontologies and Linked Data (VOILA '17), CEUR-WS.org, CEUR Workshop Proceedings, vol 1947

Schlegel K, Weißgerber T, Stegmaier F, Granitzer M, Kosch H (2014) Balloon Synopsis: A jQuery plugin to easily integrate the Semantic Web in a website

Skjaeveland MG (2012) Sgvizler: A javascript wrapper for easy visualization of sparql result sets. In: Extended Semantic Web Conference, Springer, pp 361–365