LinkDaViz – Automatic Binding of Linked Data to Visualizations

Klaudia Thellmann, Michael Galkin, Fabrizio Orlandi, Sören Auer

University of Bonn & Fraunhofer IAIS, Bonn, Germany
klaudia.thellmann@iais.fraunhofer.de, galkin@iai.uni-bonn.de,
orlandi@iai.uni-bonn.de, auer@cs.uni-bonn.de

Abstract. As the Web of Data is growing steadily, the demand for user-friendly means for exploring, analyzing and visualizing Linked Data is also increasing. The key challenge for visualizing Linked Data consists in providing a clear overview of the data and supporting non-technical users in finding suitable visualizations while hiding technical details of Linked Data and visualization configuration. In order to accomplish this, we propose a largely automatic workflow which guides users through the process of creating visualizations by automatically categorizing and binding data to visualization parameters. The approach is based on a heuristic analysis of the structure of the input data and a comprehensive visualization model facilitating the automatic binding between data and visualization parameters. The resulting assignments are ranked and presented to the user. With LinkDaViz we provide a web-based implementation of the approach and demonstrate the feasibility by an extended user and performance evaluation.

1 Introduction

The amount of data published as Linked Data is continuously increasing, but for end users it is still cumbersome to exploit and difficult to appraise the value of this data. A reason for this is that we still lack comprehensive means for user-friendly and engaging exploration and visualization of Linked Data.

Visualization is one of the most challenging but at the same time rewarding aspects of exploring Linked Data. We have a plethora of data modalities (factual, temporal, spatial, statistical, schema and meta data) and vocabularies for all of these. At the same time there is a vast variety of visualization and exploration techniques [4], most of which are limited either in generality [5] or usability for non-technical users [6, 10]. Thus, the key challenge of Linked Data visualization consists in hiding the technical details of Linked Data and visualization configurations and finding a balance between generality and usability. This can be accomplished by automatizing the process of producing visualizations, which would greatly facilitate the interaction with data by end users [2, 13].

In this article we present our approach LinkDaViz which supports the user in selecting and configuring visualizations by automatically binding Linked Data

to visualizations. The approach is based on an analysis of the input data structure and a comprehensive visualization model comprising structural and layout options. The binding problem between the data and the visualization is reduced to an assignment problem involving cost functions and heuristics. The resulting assignments are ranked and the highest ranking visualization instantiations presented to the user. The user can further refine and configure these visualization instances.

The contributions of this work are in particular:

- a method for visualization-oriented input data analysis aiming at discovering structures relevant for visualization,
- a formal visualization model, comprising structural and layout options,
- a visualization recommendation algorithm that automatically binds the selected data properties to visualization parameters,
- an implementation of the approach with the LinkDaViz web application.

The LinkDaViz visualization tool simplifies the interaction with datasets unknown to a user in ways unforeseen by the publisher. We demonstrate the feasibility of our approach with an extensive user study and an evaluation of scalability and effectiveness.

2 Related Work

Most existing approaches are only usable by a technical audience or limited to certain domains or data representations [4]. In order to hide complexity of data selection and visualization configuration, the focus of visualization approaches has been shifting towards automation [2, 5, 13].

Klimek et al. [6] implemented a workflow (Payola) based on the Linked Data Visualization Model (LDVM) [2], which consists of various analyzers for automatically classifying datasets and transformers for mapping the data to visualization abstractions. However, the amount of manual configuration and the necessary transformation steps between different abstractions might be considered a shortcoming by non-technical users.

Voigt et al. [14] propose a generic approach for visualization configuration in form of a faceted browser (Vizboard). The user creates a weighted query on abstract visualization features, which is used to automatically compute the visualization suggestions to be presented to the user. LinkDaViz implements a different mechanism for automatically suggesting visualization without expecting the user to know beforehand how he intends to visualize the data.

Mutlu et al. [8] developed an approach for automatically mapping data to visualizations that have a similar input structure as a given RDF Data Cube. LinkDaViz follows a similar mapping approach, but is not limited to the Data Cube vocabulary and has a more generic matching based on bipartite graph matching.

Bikakis et al. [1] propose rdf:SynopsViz, which comprises features such as on-the-fly hierarchy construction, statistics, faceted browsing and measuring data

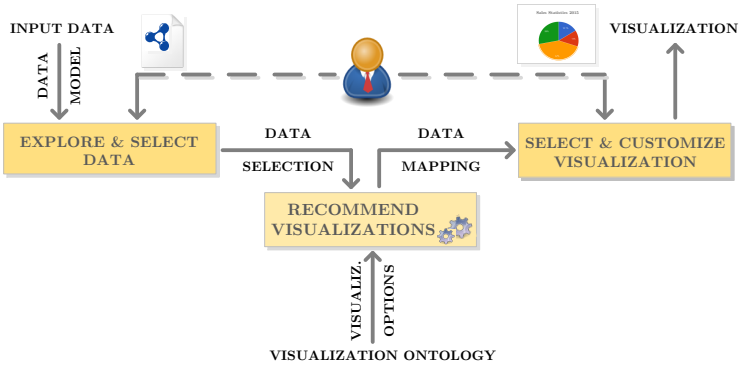


Fig. 1: The Linked Data Visualization Workflow.

quality through dataset metadata. Bindings for a user-selected visualization (five types of charts, timeline or treemap) are computed automatically based on a selection of classes and properties. LinkDaViz supports any data types (not only numeric and dates), offers selection of data properties of different nesting depth, and automatically recommends visualizations, and not only bindings to a manually selected visualization.

With LinkDaViz we aim to find a balance between generality and ease of use. Hence, we aim at improving on existing approaches and supporting the user in selecting and configuring visualizations for arbitrary Linked Data through automatic visualization recommendation and intuitive customization.

3 Approach

In this section, the workflow for visualizing Linked Data is described formally. The goal of the workflow is to support users in selecting and configuring visualizations by providing a largely automatic visualization workflow. The visualization workflow is based on the assumption that the user does not necessarily know how to choose and manually configure visualizations for a certain dataset, but can decide whether a proposed visualization configuration is reasonable. Thus, the challenge is to compute a ranked list of visualization configurations from a subset of a dataset selected by a user for its visualization. It is not in the scope of this work to guess what part of a dataset the user might be interested in, but to provide assistance in visualizing a previously specified subset of the data.

The task of finding configurations for a visualization can be modeled as an assignment problem which describes how the selected data can be mapped to the visualization's input parameters. The assignment problem can then be solved using a weighted bipartite graph matching algorithm.

3.1 Visualization Workflow

The visualization workflow (depicted in Figure 1) guides the user through the process of visualizing data and starts with the exploration of a dataset. After the user has selected the part of the dataset to be visualized, a ranked list of recommended visualizations is computed. When one of the recommendations is selected, the resulting visualization is displayed, ready for customization.

For each workflow step, a different representation of data is needed, starting with the input data model, which is structured in a tree-like fashion suited for the exploration of data. The LinkDaViz ontology contains a description of the visualization's structural and layout parameters and the scales of measurement used for specifying what types of input data can be mapped to a visualization's parameters. The purpose of the ontology is to serve as a basis for the recommendation algorithm so it provides only the information necessary for computing visualization mappings. The data selected for visualization constitutes a subset of the input data and serves, together with the visualization options extracted from the visualization ontology, as input for the recommendation algorithm.

The following subsections contain a detailed description of the data representations, the formalization of visualizations and the recommendation algorithm.

3.2 Input Data Model

The input data model is a abstract description of the input for the LinkDaViz tool and is automatically generated for the selected dataset during the data exploration and selection phase of the visualization workflow. In order to obtain the model, a tree representation of the input dataset is built automatically level by level as the user browses through the classes and their appendant properties. The input data model consists of a finite set DS of datasets and a set of trees $Trees(ds)$ for each dataset $ds \in DS$. Each tree $T \in Trees(ds)$ is a directed tree $T = (V, E, r)$ with a root node $r \in V$ corresponding to an RDF class, and is defined by a set of nodes V and a set of edges E. For instance, the dataset depicted in Figure 2 contains statistics about European countries and can be modeled as two trees T_1, T_2 with root nodes $r_1 =$ `Country` and $r_2 =$ `EU-Member`.

The set of nodes V consists of inner nodes representing RDF object properties $O \subset V$ and leaf nodes representing RDF data properties $D \subset V$:

$$V = \{r\} \dot\cup D \dot\cup O \tag{1}$$

where for every edge $(v, w) \in E$, $v \notin D$ (that is, data properties are leaves).

For instance, the data properties nodes of tree T_1 from the first level are $D_1 =$ { `Name`, `Population`, `Area`, `Code`} and one object node $O_1 = \{$`Capital`$\}$. Each data property node $d \in D$ has a scale $s_{data}(d)$ and a role $r_{data}(d)$, which are used in the computation of mappings to visualization options.

Data Scales The scales of measurement are divided into categorical (*Nominal*, *Ordinal*) and quantitative (*Interval*, *Ratio*) scales and are used to categorize

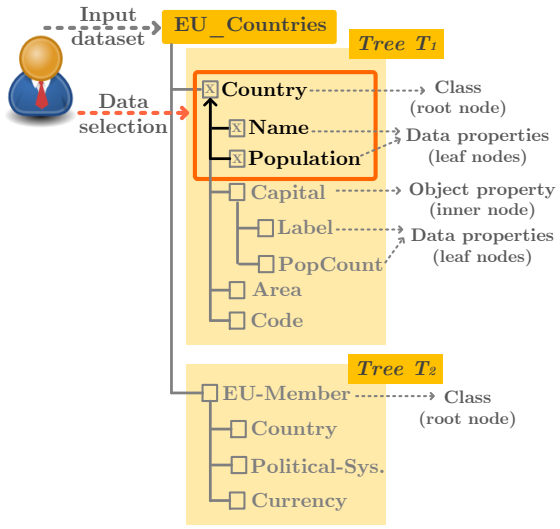


Fig. 2: Input Data Model.

data properties [11]:

$$S = \{Nominal, Ordinal, Interval, Ratio\} \tag{2}$$

The only requirement for *Nominal* data properties is to have distinguished values, for instance gender: *male* and *female*.

Ordinal data properties additionally have an order between their values, like the values *always*, *sometimes*, *never* of a Likert scale. For properties of *Ordinal* scales, no notion of numerical difference is implied, even if the values are expressed as numbers.

In case of *Interval* or *Ratio* data properties, it makes sense to compute the numerical difference between their values. The difference between *Interval* and *Ratio* is that for *Ratio* values, zero is absolute. For instance, temperature can be measured using the *Interval* scale Celsius or the *Ratio* scale Kelvin, which in contrast to Celsius has an absolute zero point.

Based on these definitions, the hierarchy of scales can be defined as follows:

$$Nominal \longleftarrow Ordinal \longleftarrow Interval \longleftarrow Ratio \tag{3}$$

with *Nominal* being the most generic and *Ratio* the most specific scale [12].

The categorization of data properties is performed heuristically depending on the data types of the data property values. Numbers are categorized as *Ratio*, dates as *Interval* and strings as *Nominal* data. For example, the scales of the data properties nodes $\texttt{Name}, \texttt{Population} \in D_1$ are $s_{data}(\texttt{Name}) = Nominal$ and $s_{data}(\texttt{Population}) = Ratio$.

Data Roles Data property nodes $d \in D$ also may have the role $r_{data}(d) \in R$ which can be that of an independent variable (*Domain*) or of a dependent

variable (*Range*):

$$R = \{Domain, Range, none\} \tag{4}$$

This is needed because due to the structure of some visualizations, only independent variables can be reasonably assigned to some of their input parameters, while for others only dependent variables are appropriate. Hence, this information can be used to exclude certain unreasonable mappings.

The roles that can be associated to the data properties `Name`, `Population` $\in D_1$ are $r_{data}(\texttt{Name}) = Domain$ and $r_{data}(\texttt{Population}) = Range$. The roles of a data property can be extracted for instance from the metadata provided by the RDF Data Cube vocabulary.

Data Selection The tree-like representation of the input data allows the user to browse the dataset and select some of the properties for visualization. The data selection $T_s = (V_s, E_s, r_s)$ is used as input for the recommendation algorithm and is a subtree of a tree $T \in Trees(ds)$ from a dataset $ds \in DS$ from the input data model, with T_s containing a subset $V_s \subseteq V$ of the nodes of T and all of the edges $E_s = E \cap V_s \times V_s$ between these, and with the same root $r_s = r \in V$ as T:

$$V_s = \{r_s\} \dot\cup D_s \dot\cup O_s \tag{5}$$

with $D_s \subset D$ and $O_s \subset O$.

For instance, the data selected from the `EU-Countries` dataset in figure 2 consists of two data property nodes from tree T_1, namely `Name`, `Population` $\in V$. This selection is a subtree of T_1 with the same root node $r_s = r_1 =$ `Country` and the edges $E_s = E_1 \cap V \times V = \{$(`Country`, `Name`), (`Country`, `Population`)$\}$.

3.3 Visualization Model

A visualization model is introduced for formally describing the options defining the structure and the layout of a visualization and a mapping specifying the configuration of the visualization options. The visualization model *VM* is composed of sets of structural and layout options *SO* respectively *LO* and a set of all possible mappings *M* of a data selection T_s to a visualization *vis*:

$$VM = (SO, LO, M) \tag{6}$$

Structural Options The purpose of the structural options $so \in SO$ of a visualization is to define the skeleton of the visualization. For instance, the structure of a column chart is defined by the following options: $SO = \{$*horizontal axis, vertical axis, groups*$\}$. A structural option $so \in SO$ is described by a name $n_{vis}(so) \in \Sigma^*$, a set of scales $s_{vis}(so) \in \mathcal{P}(S)$, a role $r_{vis}(so) \in R$ and a cardinality $c_{vis}(so) \in C$, with Σ^* being the set of strings in a given alphabet and $C = \{(0,1), (0,*), (1,1), (1,*)\}$ being the set of possible cardinalities, and S and R as in section subsection 3.2.

A structure option $so \in SO$ has a set of scales $s_{vis}(so) \subset S = \{$*Nominal*, *Ordinal*, *Interval*, *Ratio*$\}$, which represent the kinds of input data that can be

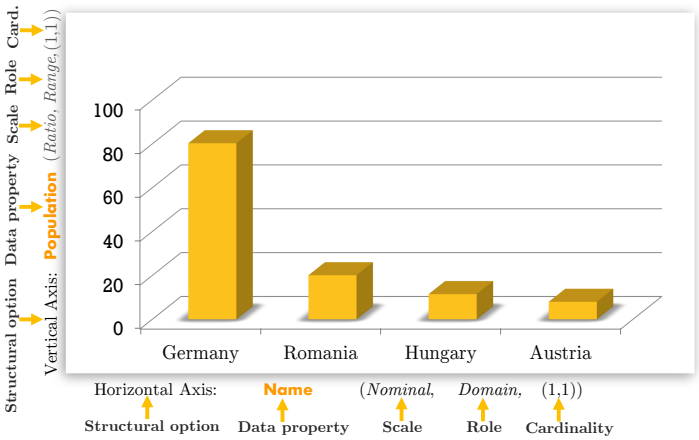


Fig. 3: Example of a structural options mapping.

mapped to it. For instance the scales of a column charts structural options are $s_{vis}(\textit{vertical axis}) = \{\textit{Interval}, \textit{Ratio}\}$, $s_{vis}(\textit{horizontal axis}) = \{\ \textit{Nominal}\}$ and $s_{vis}(\ \textit{groups}) = \{\textit{Nominal}\}$, which results in the following pattern describing the kinds of input data that can be mapped to a column chart:

$$\underbrace{\textit{horizontal axis}}_{\text{Nominal}} \times \underbrace{\textit{groups}^{?}}_{\text{Nominal}} \longrightarrow \underbrace{\textit{vertical axis}}_{\text{Ratio}} \tag{7}$$

The role $r_{vis}(so) \in R$ of a structure option is used to restrict the data properties to independent or dependent variables that can be mapped to visualization options in order to ensure that the data conforms to the structure of the visualization. $R = \{\textit{Domain}, \textit{Range}, \textit{none}\}$. For instance the roles of a column charts structural options are $r_{vis}(\textit{vertical axis}) = \{\textit{Range}\}$, $r_{vis}(\textit{horizontal axis}) = r_{vis}(\textit{groups}\ ) = \{\textit{Domain}\}$, which results in the following pattern:

$$\underbrace{\textit{horizontal axis}\ \times\ \textit{groups}^{?}}_{\text{domain}} \longrightarrow \underbrace{\textit{vertical axis}}_{\text{range}} \tag{8}$$

The roles of the selected data properties `Name` and `Population`, are $r_{vis}(\texttt{Name}) = \{\textit{Domain}\}$ and $r_{vis}(\texttt{Population}) = \{\textit{Range}\}$. This results in the following distribution of roles:

$$\underbrace{\textit{horizontal axis}}_{\text{domain = Name}} \longrightarrow \underbrace{\textit{vertical axis}}_{\text{range = Population}} \tag{9}$$

Which means that each European country has a population count as value, which is projected as a vertical bar in a column chart.

Finally, the cardinality $c_{vis}(so) \in C$ indicates if a structure option is required or optional and if it is single or multi-valued. The cardinality is indicated by affixing ? for cardinality $(0, 1)$, $*$ for $(0, *)$ and $+$ for $(1, *)$ to the

scale of measurement, with no affix indicating cardinality $(1,1)$. For instance, the cardinalities of the structural options of column charts are: $c_{vis}(\textit{horizontal axis}) = c_{vis}(\textit{groups}) = (1,1)$ and $c_{vis}(\textit{vertical axis}) = (1,1)$, which results in the pattern:

$$\underbrace{\textit{horizontal axis}}_{\text{Required}} \times \underbrace{\textit{groups}^{?}}_{\text{Optional}} \longrightarrow \underbrace{\textit{vertical axis}}_{\text{Required}} \tag{10}$$

Layout Options The layout options LO allow the user to refine the style of the visualization. As they don't have any influence on the recommendation, they are not formally described in detail. For instance, a column chart's layout can be specified by the following options: $LO = (\textit{number grid lines, v-axis label, h-axis label, stacked columns})$

Options Mapping An options mapping $m = (m_{SO}, m_{LO})$ of the data selection T_s to the visualization is composed of a structural options mapping $m_{SO} \subseteq SO \times D_s$ assigning data property nodes to structural options and a layout options mapping $m_{LO} = \{(lo_1, \textit{value 1}), (lo_2, \textit{value 2}), \ldots, (lo_m, \textit{value m})\}$ assigning values to layout options $lo_i \in LO$ for $i \in \{1, \ldots, m\}$, $m \in \mathbb{N}$. The structural options mapping must conform to the cardinality constraints of each structural option as explained above. Initially, a computed suggestion can be assigned to the structural options, and predefined default values from the visualization ontology to the layout options.

An example of a possible mapping of the selected data properties from the `EU-Countries` example in subsection 3.2 to a column chart's structural options would be $m = (\{(\textit{vertical axis}, \texttt{Population}), (\textit{horizontal axis}, \texttt{Name})\}, \{(\textit{number grid lines}, 9), (\textit{v-axis label}, \texttt{Population Count}), (\textit{h-axis label}, \texttt{Country}), (\textit{stacked columns}, \texttt{false})\})$. This mapping is depicted in Figure 3 and results in a statistical visualization of the population count of European countries. Because no grouping values were specified the optional structural option *groups* is left out.

3.4 Visualization Recommendation

In order to suggest possible visualization configurations, a recommendation algorithm has been introduced, which automatically binds the selected data properties to visualization options. For each visualization, the assignment of data properties to the visualization's structural options is modelled as a weighted bipartite graph matching problem and solved by the graph matching algorithm introduced by Kuhn and Munkres [3, 7]. The optimal solutions of these graph matching problems are then ranked to produce a list of recommended visualizations.

Assignment Problem The assignment problem for a visualization is defined as follows: Given the set $D_s = \{d_1, d_2 \ldots d_m\} \subset V_s$ of data properties from the data selection T_s and the set of structural options $SO = \{so_1, so_2 \ldots so_n\}$, find a valid structural options mapping m_{SO} with maximum possible number of assigned structure options and minimal cost.

The lower the cost of a mapping, the better it conforms to the input data the visualization expects, and thus the higher the likeliness that it corresponds to what the user envisioned. A high number of assigned data properties ensures that a large part of the selected data can be visualized.

The cost w_{ij} of mapping the data property d_j to the structural option so_i is the sum of three penalties, namely scales, roles and optionals penalty, which are introduced in more detail below.

$$w_{ij} = w_{ij}^{scale} + w_{ij}^{opt} + w_{ij}^{role} \tag{11}$$

Scales Penalty The amount of the scales penalty w_{ij}^{scale} as displayed in Figure 4*a* indicates how well a data property's scale $s_{data}(d)$ matches a structural options's scales $s_{vis}(so)$. The higher the penalty the more unfitting the match of the data property d and the structural option so. When $s_{data}(d)$ is contained in $s_{vis}(so)$ this indicates a perfect match, so $w_{ij}^{scale} = 0$. In contrast, an invalid match is given a high penalty $w_{ij}^{scale} = w_{inv}$, for example $w_{inv} = 1000$.

The values of the penalties are chosen in a way to increase the likeliness of computing meaningful mappings.

For instance, it is more appropriate to map an *Interval* data property to a *Nominal* structural option than a *Ratio* property, because less information is lost for the *Interval* property, as *Interval* is more general than *Ratio* (cf. subsection 3.2).

		Structural Option		
Scales Penalty		*Nominal*	*Interval*	*Ratio*
Data Property	*Nominal*	0	w_{inv}	w_{inv}
	Interval	30	0	w_{inv}
	Ratio	40	30	0

(a) Scales penalties overview.

		Structural Option		
Role Penalty		*Domain*	*Range*	-
Data Property	*Domain*	0	w_{inv}	20
	Range	w_{inv}	0	20
	-	20	20	0

(b) Roles penalties overview.

	Structural Option	
Optional Penalty	*Optional*	*Required*
	50	0

(c) Optionals penalties overview.

			Structural Option		
Cost Matrix			so_1	so_2	so_3
			horizontal axis	*vertical axis*	*groups*
Data Property	d_1	*Name*	0	1000	50
	d_2	*Population*	1000	0	50

(d) Cost matrix example.

Fig. 4: Penalties overview and cost matrix example.

Roles Penalty The roles penalty w_{ij}^{role} is added if the structural option has a defined role and the data property's role is not known, which might result in an incorrect match, or if the option has no role specified but the data property's role is known (see Figure 4*b*). In the latter case, it might be beneficial to favor mapping properties to the options whose roles are known by penalizing the

mappings to the options with no associated roles. In both cases the role penalty has a higher significance than the scale penalty: $w_{ij}^{role} < w_{ij}^{scale}$ for non-perfect mappings ($w_{ij}^{scale} > 0$). In the cases in which either both sides' roles are known or not known, no role penalty is added.

Optionals Penalty In case of optional structural options, an optionals penalty w_{ij}^{opt} is introduced to ensure that required structural options are preferred in the mapping, thus reducing the likeliness of producing an invalid mapping (see Figure 4 *c*). Therefore, the optional penalty must be greater than the greatest scale penalty of valid assignments (which is 40): $w_{ij}^{opt} = 50 < w_{ij}^{scale}$ for valid mappings ($w_{ij}^{scale} < w_{inv}$).

Cost Matrix In order to solve an assignment problem a cost matrix $\mathcal{W} = (w_{ij})_{i=1\ldots n, j=1\ldots m}$ is computed for all available visualizations by adding the scale penalty w_{ij}^{scale}, the optional penalty w_{ij}^{opt} and the role penalty w_{ij}^{role}. For the formal description, a square matrix is more favorable, which can be achieved by padding the weight matrix with zeroes (dummy values) [3]. From here on, it is assumed that the matrix is square: $n = m$

The cost matrix $\mathcal{W}$ for computing the mapping $m_{SO} = \{$(*vertical axis*, `Population`), (*horizontal axis*, `Name`)$\}$ of a column charts structural options $SO =$ (*horizontal axis, vertical axis, groups*) and the data selection $D = \{$`Name`, `Population`$\}$ from the `EU-Countries` is depicted in Figure 4 *d*.

The column entries from each row are composed of the sum of the scale weight, role and optional penalty. For instance, the mapping of data property $d_1 =$ `Name` $\in D$ to the structural option $s_{vis}(s_1) =$ *horizontal axis* has the cost $w_{11} = w_{11}^{scale} + w_{11}^{role} + w_{11}^{opt} = 0$. Because the scales $s_{data}(d_1) = s_{vis}(s_1) =$ *categorical* and the roles $r_{data}(d_1) = r_{vis}(s_1) =$ *domain* match, no penalties are added. As the structural option cardinality $c_{vis}($*horizontal axis*$) = (1, 1)$ indicates that this option is required, no optionals penalty is added.

Mapping Given the cost matrix $\mathcal{W} = (w_{ij})_{i=1\ldots n, j=1\ldots n}$ as input for the bipartite graph matching algorithm a maximal mapping from the selected data T_s to the structural visualization options *SO*, is computed. That is a permutation π of $\{1, 2, \ldots, n\}$ is determined with maximum value of

$$w_{m_{SO}} = \sum_{i=1}^{n} w_{i,\pi(i)} \tag{12}$$

Ranking After computing a mapping for each visualization, the mappings are ranked and the highest ranking one is presented to the user along with the other recommendations as alternative visualizations.

A mapping m_{SO} is excluded from the result list if its cost is higher than a threshold $w_{m_{SO}} \geq w_{inv}$ or if it has unassigned required structural options: $\nexists$ *data property* such that $\nexists($*so*$_i$, *data property*$) \in m_{SO}$ and $c^v($*so*$_i) \in \{(1, 1), (1, *)\}$.

A mapping m_{SO_1} is ranked higher than a mapping m_{SO_2} if m_{SO_1} has a larger number of assigned structural options than m_{SO_2}, that is, $|m_{SO_1}| > |m_{SO_2}|$ and if m_{SO_1} has a lower cost than m_{SO_2}: $w_{m_{SO_1}} < w_{m_{SO_2}}$.

4 Implementation

Architecture In order to guide the user through the process of visualizing Linked Data, a JavaScript based web-application, LinkDaViz[1], has been developed. The application receives data in RDF or tabular format as input. It consists of a frontend module for exploring and selecting data and configuring visualizations, and a back-end module for computing visualization recommendations (see Figure 5). The frontend module is realized using *Ember.js*[2], an open-source JavaScript

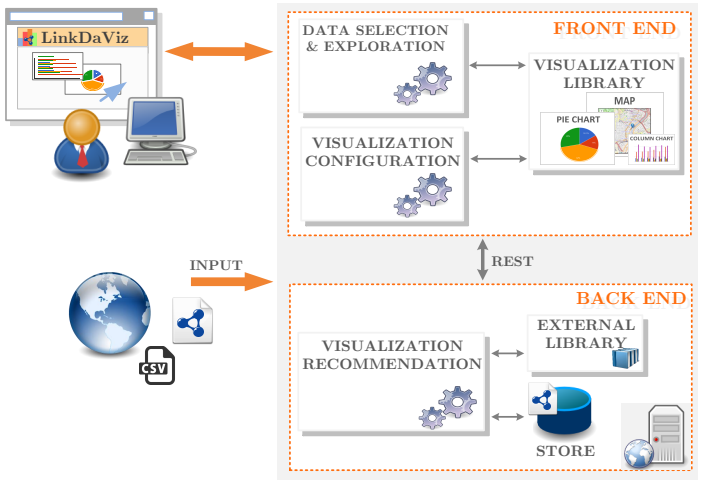


Fig. 5: Architecture of LinkDaViz.

client-side MVC framework, and consists of a component for querying and categorizing data, the visualization widgets library and a component for configuring a visualization. The selected input dataset is queried and categorized by determining the scale and role of each data property. The visualization widgets library contains configurable visualizations for statistical, temporal and geographical data (e.g. charts based on *D3/Dimple*[3], maps based on *Leaflet*[4]) and visualizations for previews (e.g. tables). The component for configuring visualizations is in charge of initializing and triggering the rendering of the recommended visualizations.

The backend is written in JavaScript and runs on *Node.js*[5], an open-source runtime environment for server-side applications, and consists of: *i)* a component

[1] Publicly available at: http://eis.iai.uni-bonn.de/Projects/LinkDaViz.html
[2] http://emberjs.com/
[3] http://d3js.org/, http://dimplejs.org/
[4] http://leafletjs.com/
[5] https://nodejs.org/

for computing visualization recommendations and *ii)* a component managing the data being queried by the frontend through a REST API. The store contains datasets, the visualization ontology, from where the visualization metadata is extracted and the saved visualization configurations that can be reloaded for further customization.

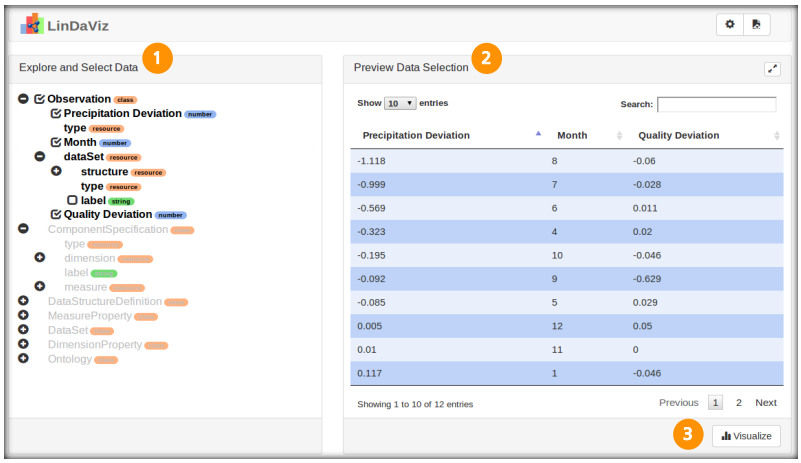


Fig. 6: LinkDaViz - select data: **1.** Browse and select data. **2.** Explore selection. **3.** Visualize selection.

Data exploration and selection The component for data selection displayed in Figure 6 of the LinkDaViz UI consists of a tree-view of the input dataset with labeled nodes for browsing and selecting data (Figure 6 **(1)**) and a preview for exploring the selection (Figure 6 **(2)**). As described in subsection 3.2, the input dataset is modeled as a list of trees, with RDF classes as root nodes. Each tree consists of a set of inner nodes representing RDF object properties and leaf nodes representing RDF data properties. Data properties are labeled corresponding to the data type of their values (e.g. number, string, date, spatial) and object properties are labeled as resources. When selecting data properties, a preview is generated displaying the values of the data properties in a table. Following the selection step is the actual visualization of the selected data **(2)**.

Visualization selection and customization The component for selecting and customizing visualizations of the LinkDaViz UI is depicted in (Figure 7) and consists of a list of visualization suggestions that are computed based on the dataset's content (Figure 7 **(4)**), a configuration component (Figure 7 **(5)**), the visualization (Figure 7 **(6)**) and consumption actions (export, save Figure 7 **(8)**) and tuning options (Figure 7 **(7)**). The configuration component contains an overview of the selected properties and the suggested mapping ("Visualization

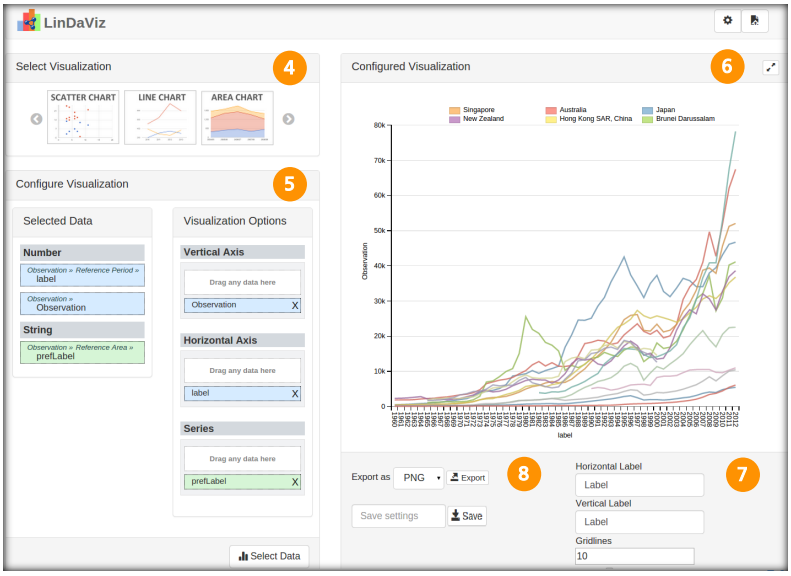


Fig. 7: LinkDaViz - visualize data: **4.** Select recommended visualization. **5.** Customize suggested configuration. **6.** Visualize data. **7.** Customize layout. **8.** Save or export visualization.

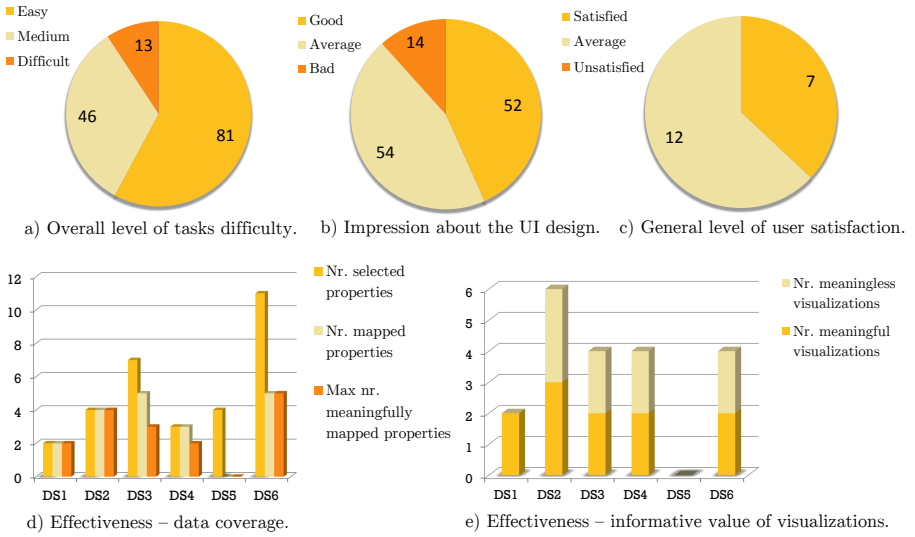
Options") for the selected visualization. The suggested mapping can be manually changed by dragging and dropping properties from the list of selected properties to the visualization options. The visualization's layout can be customized as well, for instance by adding labels to the axes or changing the number of grid lines. Finally, the visualization can be exported in different formats (e.g. PNG, SVG) and saved for later re-use.

5 Evaluation

In order to evaluate usability, scalability and effectiveness of the LinkDaViz tool, a comprehensive study has been conducted.

Setup For the evaluation the application was deployed in Linux Virtual Machine on Intel Core i5 machine (2.5 GHz, 4GB RAM) and on an Amazon EC2 t2.micro instance (2.5 GHz, 1 GB RAM). The datasets used for evaluation contain statistical, temporal and geographical data and were collected mainly from the World Bank Linked Data project, Data.Gov and DataHub.

Methodology In preparation of the user study, the participants and the evaluation criteria were identified, and a list of tasks to perform and a feedback questionnaire were created. Overall 20 participants of age range 20-30 took part in the evaluation, i.e. 15 males, 5 females, 17 students, 2 PhD students and one working professional. Seven tasks and corresponding questions were composed



a) Overall level of tasks difficulty. b) Impression about the UI design. c) General level of user satisfaction.

d) Effectiveness – data coverage. e) Effectiveness – informative value of visualizations.

Fig. 8: Evaluation Results of LinkDaViz.

in order to evaluate the level of difficulty, the UI design, the effectiveness of the produced visualizations and the scalability of the recommendation engine. The tasks the users were asked to perform have been designed according to standard user evaluation protocols [9] and are the following: 1) Select a dataset, 2) Assign parameters of a visualization, 3) Visualize data, explore a chart, 4) Modify structural options of a chart, 5) Modify layout options of a chart, 6) Save a chart, 7) Visualize another slice of a dataset.

To evaluate effectiveness aspects of the recommendations, the number and quality of suggested visualizations were measured by exploring and estimating possible visualizations and configurations of visualizations for a particular dataset. The quality of recommendations was determined according to how much of the selected data could be visualized and on the meaningfulness of the suggested visualizations from the perspective of the evaluators.

Scalability was tested by producing visualizations for datasets of different sizes. The largest dataset consists of about 350 millions triples.

Results In figure Figure 8*a* and Figure 8*b* the number of responses for each question regarding the level of difficulty and the UI design are summed up. From the 20 participants who took part in the evaluation, we collected 140 (Figure 8*a*) responses to the seven questions on difficulty and 120 (Figure 8*b*) responses to the six questions on the UI design. As Figure 8*a* indicates, the overall impression of performing the tasks was rated satisfactorily as average to good. The majority of the participants experienced little difficulties with the tasks given and considered the LinkDaViz UI as an effective and easy to use application for selecting data and configuring visualizations. The design of the UI was also rated average to good (Figure 8*b*). LinkDaViz can successfully produce previews and

visualizations of RDF and CSV datasets in the majority of cases, depending on the quality (e.g. correct datatypes), nature and size of the selected data. The overall level of satisfaction (Figure 8*c* – one person didn't answer the question) and the quality of recommendations was rated positively. The quality varied with the number of selected data properties that could be automatically visualized (Figure 8*d*) and the meaningfulness of the suggestions (Figure 8*e*). Having conducted the effectiveness evaluation on six datasets (DS1 to DS6) collected from Data.Gov it should be noted that there is a negative impact on recommendations from the varying dataset quality (e.g. wrong datatypes, non-uniform properties, aggregated values represented like raw data). A visualization has been generated and displayed for almost every dataset in a reasonable time. However on a selection from a large, homogeneous dataset with more than 300 millions of triples no visualization could be produced due to the browsers' memory limit.

6 Conclusion and Future Work

In this paper we have introduced LinkDaViz, a novel visualization approach and software implementation which allows for automatic visualization of Linked Data. A formal description of the approach, the visualization components and the input data has been provided. Additionally, we have introduced a recommendation algorithm that automatically binds selected data properties to visualization options. The validity of our approach has been tested with a comprehensive evaluation of our LinkDaViz software tool. We conducted a user study to evaluate usability, effectiveness and scalability of the tool. The results of the evaluation are very encouraging, considering the complexity of the task of automating a typical visualization workflow.

The insight we gained through the evaluation is that we managed to develop an easy to follow workflow for creating visualizations for RDF data and that the suggested visualizations were indeed helpful for the participants. However, the meaningfulness of the recommendations varied depending on the selected subset of data and the data quality. One possible improvement would therefore be to provide assistance to the user not only in choosing and configuring visualizations but also in choosing a reasonable subset of data that can be visualized.

Acknowledgments

The research and implementation of this work has been conducted by the University of Bonn within the LinDA project, funded by the European Union's Seventh Framework Programme under grant agreement no 610565 and at Fraunhofer IAIS within the European Union's Horizon 2020 research and innovation programme under grant agreement no 644564 for the project BigDataEurope (`http://www.big-data-europe.eu/`).

References

1. Bikakis, N., Skourla, M., Papastefanatos, G.: rdf:synopsviz - a framework for hierarchical linked data visual exploration and analysis. In: 11th Extended Semantic Web Conference (ESWC '14) (2014)
2. Brunetti, J.M., Auer, S., García, R., Klímek, J., Nečaský, M.: Formal linked data visualization model. In: Proc. IIWAS. pp. 309–318. IIWAS '13, ACM, NY (2013)
3. Burkard, R., Dell'Amico, M., Martello, S.: Assignment Problems, Revised Reprint:. Other titles in applied mathematics, Society for Industrial and Applied Mathematics (SIAM) (2009)
4. Dadzie, A.S., Rowe, M.: Approaches to visualising linked data: A survey. Semantic Web 2(2), 89–124 (2011)
5. Höfler, P., Mutlu, B.: Code query wizard and vis wizard: Supporting exploration and analysis of linked data. ERCIM News (96) (2014)
6. Klímek, J., Helmich, J., Nečaský, M.: Payola: Collaborative linked data analysis and visualization framework. In: ESWC 2013 Satellite Events, pp. 147–151 (2013)
7. Munkres, J.: Algorithms for the assignment and transportation problems. Journal of the Society for Industrial and Applied Mathematics 5(1), 32–38 (1957)
8. Mutlu, B., Hoefler, P., Sabol, V., Tschinkel, G., Granitzer, M.: Automated visualization support for linked research data. In: Lohmann, S. (ed.) CEUR Workshop Proceedings. vol. 1026, pp. 40–44. CEUR-WS.org (2013)
9. Nielsen, J., Landauer, T.K.: A mathematical model of the finding of usability problems. In: Proceedings of the INTERACT'93 and CHI'93 conference on Human factors in computing systems. pp. 206–213. ACM (1993)
10. Skjaeveland, M.G.: Sgvizler: A javascript wrapper for easy visualization of sparql result sets. In: Simperl, E., Norton, B., Mladenic, D., Della Valle, E., Fundulaki, I., Passant, A., Troncy, R. (eds.) The Semantic Web: ESWC 2012 Satellite Events, Lecture Notes in Computer Science, vol. 7540, pp. 361–365. Springer Berlin Heidelberg (2015), http://dx.doi.org/10.1007/978-3-662-46641-4_27
11. Stevens, S.S.: On the theory of scales of measurement. Science 103, 677–680 (1946)
12. Stevens, S.S.: Measurement. In: Maranell, G.M. (ed.) Scaling; a Sourcebook for Behavioral Scientists. Aldine Publishing Company (1974)
13. Voigt, M., Pietschmann, S., Grammel, L., Meißner, K.: Context-aware recommendation of visualization components. In: 4th Int. Conf. on Information, Process, and Knowledge Management, eKNOW 2012. pp. 101–109 (2012)
14. Voigt, M., Pietschmann, S., Meißner, K.: A semantics-based, end-user-centered information visualization process for semantic web data. In: Semantic Models for Adaptive Interactive Systems, pp. 83–107. Springer (2013)