RESEARCH-ARTICLE

LD viewer - linked data presentation framework

DENIS LUKOVNIKOV, University of Leipzig, Leipzig, Sachsen, Germany

CLAUS STADLER, University of Leipzig, Leipzig, Sachsen, Germany

JENS LEHMANN, University of Leipzig, Leipzig, Sachsen, Germany

Published: 04 September 2014

SEM '14: SEMANTiCS 2014 - 10th International Conference on Semantic Systems
September 4 - 5, 2014
Leipzig, Germany

LD Viewer - Linked Data Presentation Framework

Denis Lukovnikov
University of Leipzig
lukovnikov@informatik.uni-leipzig.de

Claus Stadler
University of Leipzig
cstadler@informatik.uni-leipzig.de

Jens Lehmann
University of Leipzig
lehmann@informatik.uni-leipzig.de

ABSTRACT

With the growing interest in publishing data according to the Linked Data principles, it becomes more important to provide intuitive tools for users to view and interact with those resources. The characteristics of Linked Data pose several challenges for user-friendly presentation of information. In this work, we present the LD Viewer as a customizable framework that can easily be fitted for different datasets while addressing Linked Data presentation challenges. With this framework, we aim to provide dataset maintainers with easy means to expose their RDF resources. Moreover, we aim to make the interface intuitive and engaging for both expert users and lay users.

1. INTRODUCTION

The Linked Data principles provide guidelines for publishing structured data in order to make it easily accessible to both machines and humans. In contrast to the common natural language representation of information on the Web, dereferencing Linked Data resources in many cases does not provide an intuitive view for humans. The fine granularity of triple-based data representation standardized by the RDF format may often require significant mental effort to find and integrate useful information about an entity. Moreover, some entities have a significant amount of information associated with them and users trying to access such entities may be overwhelmed by the number of shown facts.

This work presents LD Viewer, a customizable framework and resource browser for Linked Data. The LD Viewer framework aims to make it easier to address Linked Data presentation problems mentioned above by providing a powerful feature set to endpoint maintainers and requiring minimal effort to adapt the default interface for another dataset.

SEM '14, September 04 - 05 2014, Leipzig, AA, Germany

ACM 978-1-4503-2927-9/14/09...$15.00.
http://dx.doi.org/10.1145/2660517.2660539

LD Viewer is based on the DBpedia Viewer interface [11] further generalizing it by striving for higher architectural consistency. The DBpedia Viewer interface was originally only capable of presenting data from the DBpedia [9] dataset. As the DBpedia Viewer, the LD Viewer aims to present information in an engaging way while showcasing the Linked Data philosophy. LD Viewer further improves upon the Triple Action Framework (TAF) from DBpedia Viewer. The TAF allows for both easy high-level interface customization and adding interactivity to the presented triples.

The LD Viewer has been tested on several DBpedia datasets and related projects, such as LinkedGeoData. Throughout this paper, we use DBpedia and LinkedGeoData as example datasets.

The remainder of the article is structured as follows: Section 2 provides an overview of the example datasets DBpedia and LinkedGeoData. The LD Viewer user interface is described in Section 3 and the system architecture is presented in Section 4. In Section 5, we describe configuration and extensions of the system. Implementation concerns are covered in Section 6. Section 7 elaborates on related work. Some preliminary evaluation results are given in Section 8 and we conclude with Section 9.

2. DBPEDIA AND LINKEDGEODATA

DBpedia is a pioneer project in Linked Data publishing. It was one of the first Linked Open Data datasets available in 2007 and is a hub in the Linked Open Data cloud[1]. The data in DBpedia originates from Wikipedia and is extracted using the DBpedia extraction framework. The latest DBpedia release provides data for 4.0 million *entities* out of which 3.2 millions are classified according to the DBpedia ontology.[2]

DBpedia provides different kinds of information about entities. Entities typically have types, labels, links, Linked Data links and textual descriptions associated with them. DBpedia contains links to equivalent entities in other datasets (such as YAGO) and links to the same entities in DBpedia datasets for other languages (such as nl.dbpedia.org). In addition to the general information about entities, e.g. types

[1]Linking Open Data cloud diagram, by Richard Cyganiak and Anja Jentzsch. `http://lod-cloud.net/`
[2]`http://blog.dbpedia.org/2013/09/17/dbpedia-39-released-including-wider-infobox-coverage\-additional-type-statements-and-new-yago-and-wikidata\-links/`

and categories, DBpedia contains properties and classes specific for particular domains. For example, the entity (of type Person) `dbpedia:Barack_Obama` has a property `dbo:spouse` which refers to the entity `dbpedia:Michelle_Obama`.

Since the start of the DBpedia project, several tools and services were developed around DBpedia. DBpedia Spotlight [12] performs Entity Linking (or Named Entity Resolution and Disambiguation, NERD) in text by linking entity mentions to DBpedia entities. DBpedia Lookup[3] is an additional service which allows to search for DBpedia entities using either strings or prefixes (for auto-completion). The *DBpedia mappings wiki*[4] is an effort to crowdsource the mappings between the Wikipedia infoboxes and the DBpedia ontology. Apart from DBpedia specific tools, external vizualization tools have been developed that use DBpedia, such as RelFinder [7] and LodLive [5]. RelFinder explores the knowledge graph to find paths between two entities. LodLive provides a visually appealing way to explore information associated with an entity.

Such tools and services exist independently, but together form an eco system, which provides added value for the DBpedia datasets. Our new interface integrates several tools in a generic manner – partly to increase user-friendliness and partly to showcase the achievements in the Linked Data space obtained so far.

LinkedGeoData[5] (LGD) is an effort to add a spatial dimension to the Web of Data [15]. In this project, tools for RDFizing open spatial datasets are created. These datasets are then interlinked and published as Linked Data, via SPARQL endpoints and as downloads. The flagship dataset is the RDF version of the data from the OpenStreetMap (OSM) project[6], which acts as the linking hub for additional datasets, similar to DBpedia. At present, interlinking with DBpedia and GeoNames[7] is provided, as well as an integration of the datasets from *Global Administrative Area*[8] and *Natural Earth*[9]. So far, LGD's Linked Data view only offered the basic HTML pages provided by Pubby (see Section 7). With this work, users of LGD will now find a modern user interface with new features, such as a view which depicts the locations and areas of resources on a map.

3. LD VIEWER USER INTERFACE

We first present the features built into the default LD Viewer user interface. The most prominent interface element is the property table (right under the map in Figure 1), which displays all the available properties of the viewed resource, as is common among Linked Data browsers. The property table displays both forward and reverse properties and provides pagination support for reverse properties with a large number of values.

[3] `http://wiki.dbpedia.org/lookup/`
[4] `http://mappings.dbpedia.org`
[5] `http://linkedgeodata.org`
[6] `http://openstreetmap.org`
[7] `http://www.geonames.org/ontology/documentation.html`
[8] `http://www.gadm.org/`
[9] `http://www.naturalearthdata.com/`

We attempt to improve the readability of the property table by fetching and displaying labels instead of resource URI's. This is similar to the use of Fresnel lenses [13] in some other browsers (such as [2]). Label fetching is optional as it can increase the load on the triple store. In the following subsections, we discuss other features of the interface. The first eight subsections (Section 3.1 - Section 3.8) correspond to the eight features indicated in Figure 1.

3.1 Pretty Box

The *pretty box* (part one of Figure 1) displays important properties of the viewed entity. There is a predefined set of facts we provide, namely: (1) a picture, (2) the title, (3) the types, (4) a short description and (5) links to other resources. Additionally, a list of important properties of the viewed resource can be displayed. These data are generated from the set of triples describing the viewed resource using predefined mappings. The DBpedia datasets provide most of this general information for all entities.

The LD Viewer does not perform automatic selection of relevant properties to display. This is out of the scope of this project as we solely focus on customizable presentation of information.

3.2 Search Bar

The basic search functionality provided is based on SPARQL queries that use the label properties of resources to retrieve a list of search results. However, this basic search can easily be replaced by using external search services based on the dataset. For example, the DBpedia Lookup service is used with the DBpedia dataset for fast resource retrieval.

3.3 Language Filtering

The language filtering system allows the user to choose a preferred display language. This filters all literal values based on the user preferences and displays only the relevant values. In the case a literal does not exist in the preferred language, a fallback language (usually English) is chosen by default. This feature is helpful on multilingual datasets such as DBpedia where labels and abstracts exist in 12 different languages.

3.4 Triple Filtering

Part four of Figure 1 highlights the triple filtering feature. Triples can be filtered on the labels of both properties and values. This is useful for the users who quickly want to find specific properties and values. The filtering is based on string matching and supports all literal values as well as URIs.

3.5 Shortcut box

The shortcut box (part 5 in Figure 1) provides anchor links to some important properties of entities. However, the list of properties is currently hardcoded and contains links to categories, types, external links, etc...

3.6 Live Previews

When the user hovers over a link (URI, ontology property or class) a concise, language-filtered preview is displayed. For entities, this preview contains a picture (if available), the

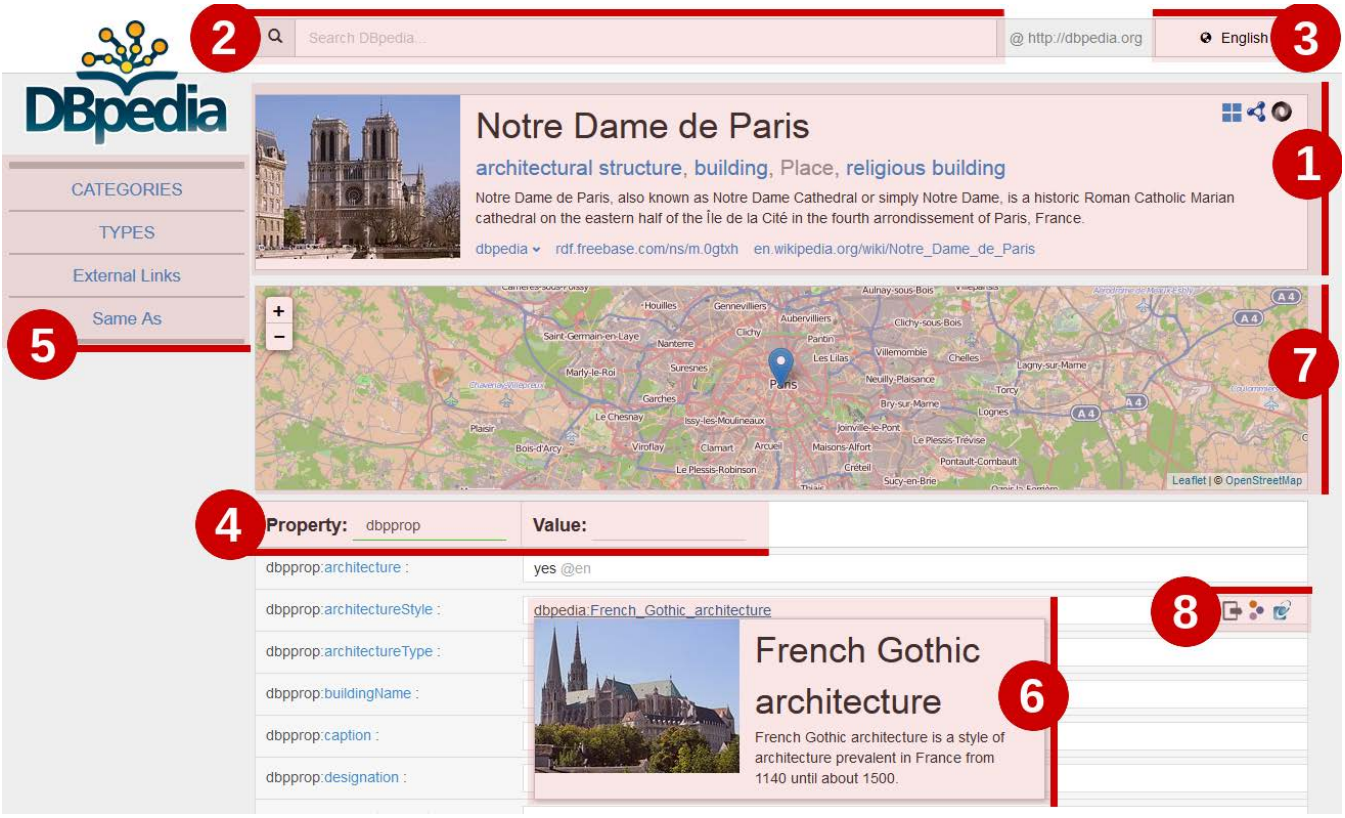

Figure 1: Screenshot of the new interface. The transparent red areas highlight new features, with the associated number in the red circle corresponding to its subsection number within ??. A quick overview: (1) pretty box, (2) search bar, (3) language switcher, (4) triple filter, (5) shortcuts, (6) preview box, (7) map and (8) triple actions.

title and a short description. Part 6 of Figure 1 shows a preview of the French Gothic architecture entity.

3.7 Maps

For entities having location information (latitude and longitude), a map is shown with its coordinates. OpenStreetMap is used for the map display.

3.8 Triple Actions

As displayed in part 8 of (Figure 1), next to each triple, different icons exist, each representing a different *triple action*. Triple actions are enabled using conditions on the triple. Thus, the set of available actions for different triples may be different. When the conditions are met, the action icon is displayed next to the triple. When the user clicks on the triple action icon, the action is executed. An elaborate discussion of triple actions and TAF as well as their use beyond adding interactivity is given later (Section 4.3).

Below is an overview of the currently implemented user actions for the DBpedia dataset:

- Annotation – uses DBpedia Spotlight to annotate text. Only applicable to texts of certain length.
- RelFinder – links to RelFinder, where the connections (including indirect ones) between the viewed entity and the value entity can be explored. Only applicable to DBpedia resources.
- LodLive – opens the value entity with the LodLive browser. Only applicable to DBpedia resources.
- OpenLink Faceted Browser – view the value entity using OpenLink Faceted Browser. Only applicable to DBpedia resources.
- Wikipedia – opens the Wikipedia page associated with the value entity. Only applicable to DBpedia resources.
- DBpedia template mapping – links to the DBpedia mapping associated with the DBpedia template. Only applicable to DBpedia resources under the Wikipedia template namespace.

Additionally, the triple action framework (TAF) allows to define action groups. Action groups allow action developers to group sets of related actions. Common action applicability checking logic (*bind* semantics) shared by all group members can be put in the action group applicability checking logic. When used wisely, this can reduce page building time by reducing the number of applicability checks executed.

3.9 Faceted browsing of class instances

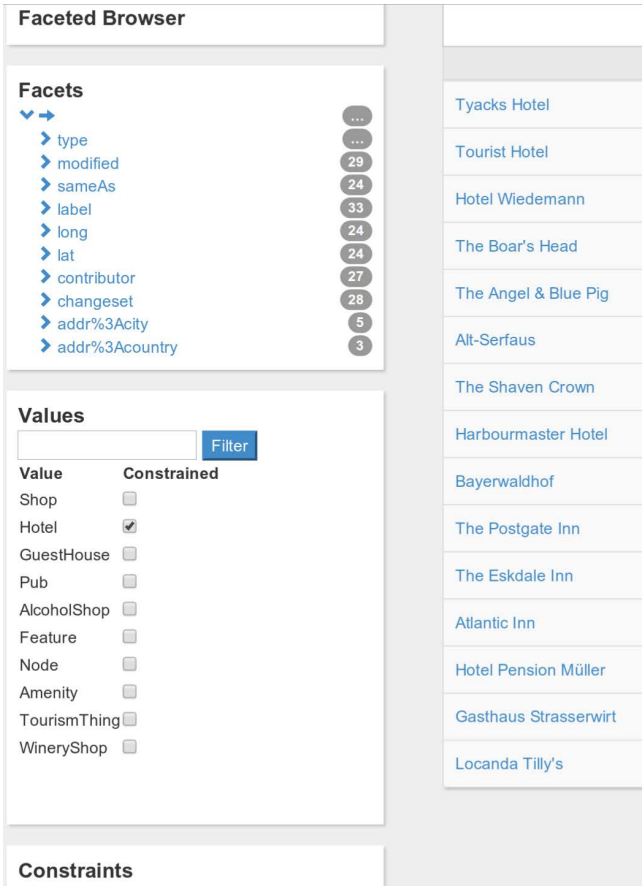

Figure 2: Screenshot showing the faceted browsing facilities for the class *Pub* at LinkedGeoData. A constraint has been set which restricts the pubs to those that are additionally typed as *Hotel*.

Ontology pages for classes offer widgets for faceted browsing of the corresponding instances, as shown in Figure 2. These widgets are provided by the *jassa-ui-angular*[10] project. The faceted browser offers a user-friendly way to define constraints on the set of instances of a class, both by immediate and indirectly related properties. In online analytical processing (OLAP) terms, these widgets enable browsing of snowflake schemas. Originally, the faceted browsing components have been developed for the spatial Linked Data browser *Facete* [16], and have recently been turned into a highly re-usable library.

3.10 Notifications and status updates

Not shown on the screenshots are the notification and status elements, which provide a means to communicate to the user what the system is currently doing (status) and bring certain messages under the user's attention (notification).

4. SYSTEM OVERVIEW

In this section we discuss the architecture of the presented framework as well as the customizability possibilities. First, the general architectural choices are discussed. This is followed by a discussion of adaptation to datasets. Finally, we specify the Triple Action Framework and explain how it powers the adaptation process.

[10] `https://github.com/GeoKnow/Jassa-UI-Angular`

4.1 General architecture

LD Viewer follows the widely used Model-View-Controller (MVC) design pattern as provisioned e.g. by the AngularJS framework. The triples associated with some resource form the Model layer in the MVC architecture. Different UI elements presented in previous sections (such as the map) belong to the View layer of the MVC architecture. Finally, the Controller layer takes care of creating and populating the View elements based on data (triples) in the Model layer and responding to user-generated events. The Triple Action Framework (TAF) provides a means to customize the Controller layer while leaving the boilerplate MVC logic (fetching and parsing data, binding events to DOM, ...) for the framework to handle.

4.2 Adapting to datasets

The heterogeneity in ontology usage of different datasets complicates the creation of generic interfaces, resulting in the need for frameworks that can easily be adapted to a certain dataset. With our framework, the primary means of customization is the Triple Action Framework, which allows the deployer to easily define custom logic without any specific knowledge on the underlying frameworks. The custom logic can be used to relay information from the triple-based model to the right interface elements, but also to define user actions. The Triple Action Framework is discussed in more detail in the next section.

4.3 Triple Action Framework

The Triple Action Framework (TAF) aims to make the integration of external tools easier and provides an easy means for adapting the framework for a certain dataset. The interface maintainer can easily add new actions or adapt existing ones for a particular deployment. TAF allows to define a triple action with the following core semantics, (1) bind and (2) execute where only the bind method is necessary.

Upon page load, for each triple, the bind method of each action is called to determine whether this action is applicable for this triple. The bind method may use any information available from the triple to decide whether the action is applicable or not. For example, the DBpedia Spotlight annotation action should only be made available for annotation of textual resources, so the bind method of this action checks whether the object of the triple is a string literal and whether it exceeds a configurable minimum length. Additionally, the bind method may also contain logic beyond applicability checking.

The execute method of an action is called when the user triggers its execution. For the DBpedia Spotlight annotation action, this method uses the text in the object of the triple, sends it to the DBpedia Spotlight API for annotation and waits for a response. When the API responds, the Spotlight action changes the display value of the object of the triple to show the annotations.

However, it is not necessary to define an execute method. An action is only bound to a triple if (1) it has a bind method

that indicates the action is applicable for this triple, (2) it has an execute method. Actions fulfilling these criteria are called *user actions* because their execution depends on them being triggered by users. User actions are represented by icons next to the triple to which they are bound.

The other kind of action is the *system action*. The execution of such actions is triggered immediately if the action is applicable to the triple. Moreover, since system actions are not available to users, they do not require an execute method. The execution logic of system actions should be defined in the bind method. The logic of system actions is executed only once and system actions are not bound to triples. An example of a system action is the selection of properties to display in the Pretty Box. This system action is called for every triple and checks whether the predicate of the triple is contained in the list of mappings defined in the action. If it encounters a match, it calls the Pretty Box element to add the property. However, the action could also make a request to some external service to dynamically determine whether the triple should be displayed in the Pretty Box.

In the actual implementation, TAF provides additional hooks, providing more functionality to define new actions with ease. Moreover, the Triple Action Framework allows to define locally/globally stateful actions.

5. CUSTOMIZATION

In interface customization, we distinguish two tasks: configuration and extension. Configuration is concerned with adapting the basic interface to some dataset. Providing additional functionality to the basic interface is the topic of extensions.

5.1 Configuration

The framework settings and the triple action framework (TAF) are the primary means of configuration of the framework. In framework settings, the datasource (endpoint, graph, ...) and global setting variables are defined. Most of these settings are used to fetch information. Using TAF, however, the dataset maintainer can quickly define new actions or change existing ones in order to customize the framework for the dataset. The system actions described above allow to populate any entity-dependent interface element. Moreover, system actions can be used to define automatically triggered events, such as redirection.

For example, consider the task of populating the list of types displayed in the pretty box. In the action definition, one can bind to specific properties of the viewed entity that define the types of that entity. and declare to set the content of the types display element to the types retrieved from properties matching the bind criteria. As an example, a simplified implementation of the type action for populating the type list in the pretty box using `rdf:type` on DBpedia data is given below (wrapping code omitted for brevity).

```
mapfrom: "rdf:type"
bind:    function(about, predicate, value){
           if (predicate.uri == this.mapfrom)
             LDViewer.addPrettyType(value) }
```

Action-based configuration provides more freedom than configuration based on lists or mappings since with the former, less assumptions are made about the kind of logic dataset maintainers require. In our case, purely declarative means of configuration (lists) would impose restrictions or require the generation of some domain-specific declarative language for advanced functionality. In contrast, imperative (action-based) configuration relieves framework developers from trying to foresee and support all possibilities while leaving the maintainer the freedom to do whatever he/she desires. For example, one could also use the types present in the description of the viewed entity to fetch additional information in order to populate other interface elements. This would be impossible with declarative configuration without clogging declaration specifications.

However, action-based configuration requires at least basic programming knowledge from the maintainer. To make the standard configuration task as easy as possible, different UI elements provide interfaces so that the action developer does not need to know the implementation details of our framework nor of any of its dependencies.

5.2 Extension

TAF lightens adding new functionality for users by providing a useful abstraction with easy access to the triple that already takes care of displaying the actions. To create a new action, one simply needs to implement the hooks with the desired action logic. However, TAF only provides means to add triple-enabled functionality that can be communicated to the user by providing action icons next to the triple.

Adding or modifying interface elements or making changes to the core behavior of the interface requires understanding of our framework and some of its dependencies. However, since the LD Viewer was implemented using recent front-end technology (AngularJS[11]), front-end developers familiar with JavaScript and MV* frameworks can easily get started and easily add their own (Angular) modules (containing custom interface elements defined through Angular directives). However, we aim to minimize the need for such changes by providing a rich set of powerful features that encompasses most deployment needs.

6. SYSTEM IMPLEMENTATION

The LD Viewer is implemented as a client-side single-page web application aimed at the latest versions of Internet browsers. Since it is a client-side JavaScript application, it is only usable by user agents that support JavaScript. For user agents without such support, any backend library for displaying triples can be used and integrated with our framework. Our framework is built using the open-source AngularJS MVC framework for JavaScript and reuses components from the Jassa library. AngularJS provides a powerful implementation of the Model-View-Controller pattern for browser applications and is widely known in the front end developer community. The Jassa library provides Jena-like services for JavaScript, allowing for easy manipulation of SPARQL queries and RDF, thus increasing maintainability of the interface. Additionally, Jassa provides Facete tools, which provide faceted browsing functionality.

[11] `http://angularjs.org/`

The framework is not bound to any specific backend or triple store. However, we implemented some optional optimizations specific for Virtuoso triple stores.

Conceptually, there are three levels when the framework is deployed for a certain dataset.
The *base level* is the triple store, which is accessible using the SPARQL query language. Any triple store supporting the SPARQL standard can be used by our framework.
The *web server* and server-side code can built on top of the database. The server simply has to serve the JavaScript application.
The third level is the *client-side* code. Our framework is a pure JavaScript application that runs on the user's browser. Once the application is loaded, the client-side code takes care of querying the triple store and building the page. This moves the page construction load from the server to the client while providing a more seamless experience to the user.

7. RELATED WORK

First, we discuss different tools that are integrated in the new interface. This is followed by an overview of some Linked Data browsers.

Integrated tools

RelFinder [7] allows users to explore connections between multiple entities in a intuitive and interactive way. Given two entities, RelFinder shows paths in the underlying RDF graph connecting the two entities. The relationship discovery algorithm used in [7] is based on the original DBpedia Relationship Finder algorithm [10]. The search algorithm is essentially a breadth-first search algorithm with several optimizations for the problem.

DBpedia Spotlight [12] is an Entity Linking (EL) system. Given a text, the purpose of EL is to find which parts of text refer to which entities. DBpedia Spotlight performs EL with DBpedia entities. The linking approach of DBpedia Spotlight consists of three steps: (1) the spotting stage where the phrases in the text are recognized that might refer to entities, (2) the candidate selection stage where possible "meanings" of spotted phrases are generated and (3) the disambiguation stage where the best candidate entity is chosen as the meaning of the phrase.

Another tool integrated as a triple action is LodLive [5]. LodLive is an exploratory tool that allows users to browse Linked Data in an interactive way, using a dynamic visual graph. Moreover, it integrates information available across different SPARQL endpoints. This way, it aims to showcase the principles behind Linked Data.

Linked Data browsers and integrators

Dadzie and Rowe [6] performed a survey of tools for Linked Data consumption. In their review, the authors make a distinction between visualization (e.g. RelFinder[7]) and presentation (e.g. Marbles). A wide range of tools is discussed and a comparative study of their features is performed. They also make a distinction between three kinds of users: (1) tech-users, (2) domain experts and (3) lay users. One of the conclusions of their survey is that the reviewed Linked Data consumption tools are mostly oriented at tech users. Some of the tools discussed by Dadzie and Rowe [6] are also discussed in this section.

The Marbles Linked Data browser [12] is a server-side application that generates HTML from Semantic Web content using Fresnel [13] vocabularies. Marbles is used in DBpedia Mobile [3], a location-based Linked Data browser which shows locations available from DBpedia on a map with information about the location.

Pubby [13] is a server-side Java application that can be configured to use a SPARQL endpoint and publish the data behind it as Linked Data. It also provides a simple (static) HTML user interface. The Graphity project [14] provides a framework for publishing RDF data or building applications around it. LDIF [14] is a framework aiming to integrate information about entities from different datasets but does not focus on displaying data.

Tabulator [4] is a generic Linked Data browser that additionally provides analytical functionality mostly aimed at tech users. For example, it is possible to display the results of different queries on one map to compare them. Other notable features are the Timeline and Calendar views. For exploring the data, Tabulator employs the Outliner paradigm which allows for intuitive tree-oriented exploration. The exploration in Tabulator can thus be seen as the table-based equivalent of LodLive.

In contrast to existing approaches, the aim of LD Viewer is to provide a feature-rich and interactive interface that can easily be customized for different datasets. Customization in LD Viewer is imperative (TAF system actions), which gives ultimate flexibility to the developers and requires only knowledge of JavaScript. This contrasts other approaches such as X3S[17] and LESS[1] that provide mostly declarative means of customization and often rely on a (custom) templating language (such as LeTL in LESS). In contrast to most other presentation approaches, LD Viewer also focuses on interactivity and integration of other tools through TAF user actions.

8. PRELIMINARY EVALUATION

To evaluate the presented framework, we conducted a user survey. The survey targeted users of the framework on one of its deployments (a link to it was shown while browsing Linked Data resources). With this survey, we wish to find out whether the users like our interface, how intuitive they experience it.

The user survey follow a task-based paradigm.
The *first part* of the survey contains questions to build a profile of the user. We only build a very limited profile that says what the current occupation of the user is (bachelor student/master student/researcher or industrial professional) and their level of acquaintance with the Linked Data standards.

[12] `http://mes.github.io/marbles`
[13] `http://wifo5-03.informatik.uni-mannheim.de/pubby/`
[14] `https://github.com/Graphity/graphity-browser`

The second part of the surveys defines tasks to be completed. We ask the visitor to retrieve answers for several questions using the DBpedia deployment of the interface. The questions are:

q1. In which country was the spouse of Albert II of Belgium born?

q2. How many inhabitants does Maribor have?

q3. Who wrote Isaac Asimov's Utopia?

q4. How many employees does Unilever have?

q5. Does Lake Baikal have a higher surface area (m^2) than Belgium?

The third part of the survey contains questions in the form of statements assessing experiences from carrying out the task from the previous part. The statements are listed below.

s1. The look and feel is user-friendly and properly presents the information.

s2. The presentation of the information is intuitive, easy to grasp.

s3. The interface is responsive.

s4. The answers to the questions were easy to find using the interface.

s5. The entity summary (pretty box) at the top of the page contains the most important information about the viewed entity.

s6. Property and value filters in the property table (under the pretty box) are useful. (If you did not use the filters, choose "no opinion")

s7. The features of the interface are presented well, easy to find and to understand.

In the survey, the user must indicate his level of agreement with these statements, ranging from not at all to entirely, with an option stating the user has no opinion.

Results

We were only able to collect 10 results with the survey so far. 10 visitors provided complete answers in the survey. Of these 10 visitors, 6 answered all the questions in the second part correctly, while all 10 answered to three out of five question in the second part correctly.

The preliminary survey indicates visitors in general like the interface and have little trouble navigating it. The participants generally agreed with the statements about the design and the intuitiveness of information presentation. Property and value filters were overlooked by some participants while the ones who used them liked them. More disagreement between participants was caused by the statement about the entity summary (**s5**), for which the majority of participants agreed that it presents the most important information about the entity while others disagreed completely. Another significant disagreement between participants concerns the last statement, presentation of features (**s7**), which overall was positively rated but with larger deviations. The statement about the responsiveness (**s3**) caused most disagreement between participants. However, this may be attributed to the varying availability of the DBpedia endpoint behind the interface.

In general, these results indicate users like the interface. However, different features of the interface can still be improved and communicated more explicitly. Also, with the survey we failed to attract enough non-expert users. Most participants indicated using Linked Data often and consider themselves more or less knowledgeable in the underlying technology.

The survey results are given in Table 1.

Table 1: Visitor survey assessment results. For each of the assessment statements, the number of participants in some agreement class are given.

Statement	total disagreement	partial disagreement	partial agreement	total agreement	no opinion
s1.	0	1	3	6	0
s2.	0	3	3	4	0
s3.	2	0	4	4	0
s4.	0	2	3	4	1
s5.	1	1	4	3	1
s6.	0	1	1	3	5
s7.	0	3	4	3	0

9. CONCLUSION AND FUTURE WORK

The LD Viewer is a customizable modular framework for interactive, user-friendly presentation of Linked Data. Starting from the DBpedia Viewer, we further generalized the interface and enforced more modularization as well as stronger architectural consistency. The Triple Action Framework proved to be useful for defining system actions, which allow for greater and easier customization of the interface.

The interface does not try to conceal the technical philosophy behind Linked Data. Instead, it embraces the philosophy and presents the data as it is in a visually appealing fashion, highlighting the underlying ideas and demonstrating the possibilities of the integrated tools.

The LD Viewer framework provides a rich set of features (including interface elements), to tackle the need for creating additional ones. However, web developers can easily add custom interface elements and other features. The framework is implemented using well-known high-end web development technologies, allowing for a great range of powerful options.

The Triple Action Framework (TAF) introduced with DBpedia Viewer and further developed with the Linked Data Viewer presented here demonstrates a method for adding interactive functionality to Linked Data, going beyond merely serving RDF facts. Such ideas may not only inspire improvements in other Linked Data interfaces but might also

evolve to a standardized framework for human interaction with data across the Semantic Web in the future.

We plan to add triple actions for the incorporation of triple validation by the end users [8, 18]. Another triple action we are investigating is the option to automatically import DBpedia triples into WikiData.

A potential area of future research is the analysis of user behavior on the interface to produce novel Entity Summarization and Entity Ranking data and methods. Entity Summarization scores can be useful for Question Answering and Semantic Relatedness. The scores can also be used to extend the pretty box with the most important entity-specific information (e.g.: *birth place* for persons) and to compute a better list of properties for the shortcut box.

We did a limited preliminary evaluation of the user experience on the website, indicating generally positive response and providing new directions for improvement. Additionally, we aim to collect feedback from action developers and contributors and work together to improve this framework.

Acknowledgements

This work was supported by grants from the European Union's 7th Framework Programme provided for the projects LOD2 (GA no. 257943) and GeoKnow (GA no. 318159).

10. REFERENCES

[1] Sören Auer, Raphael Doehring, and Sebastian Dietzold. LESS-template-based syndication and presentation of Linked Data. In *The Semantic Web: Research and Applications*, pages 211–224. Springer, 2010.

[2] Christian Becker and Chris Bizer. marbles. `http://mes.github.io/marbles/`. [Online; accessed 09-Februari-2014].

[3] Christian Becker and Christian Bizer. DBpedia Mobile: A Location-Enabled Linked Data Browser. *LDOW*, 369, 2008.

[4] 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 *Proceedings of the 3rd International Semantic Web User Interaction Workshop*, volume 2006, 2006.

[5] Diego Valerio Camarda, Silvia Mazzini, and Alessandro Antonuccio. LodLive, exploring the Web of Data. In *Proceedings of the 8th International Conference on Semantic Systems*, pages 197–200. ACM, 2012.

[6] Aba-Sah Dadzie and Matthew Rowe. Approaches to visualising linked data: A survey. *Semantic Web*, 2(2):89–124, 2011.

[7] Philipp Heim, Steffen Lohmann, and Timo Stegemann. Interactive Relationship Discovery via the Semantic Web. In *Proceedings of the 7th Extended Semantic Web Conference (ESWC 2010)*, volume 6088 of *LNCS*, pages 303–317, Berlin/Heidelberg, 2010. Springer.

[8] Dimitris Kontokostas, Amrapali Zaveri, Sören Auer, and Jens Lehmann. Triplecheckmate: A tool for crowdsourcing the quality assessment of linked data. In *Proceedings of the 4th Conference on Knowledge Engineering and Semantic Web*, 2013.

[9] Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N. Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick van Kleef, Sören Auer, and Christian Bizer. DBpedia - a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web Journal*, 2014.

[10] Jens Lehmann, Jörg Schüppel, and Sören Auer. Discovering Unknown Connections - the DBpedia Relationship Finder. In *Proceedings of the 1st Conference on Social Semantic Web (CSSW 2007)*, volume 113 of *LNI*, pages 99–110. GI, 2007.

[11] Denis Lukovnikov, Claus Stadler, Dimitris Kontokostas, Sebastian Hellmann, and Jens Lehmann. Dbpedia viewer-an integrative interface for dbpedia leveraging the dbpedia service eco system. In *Proceedings of the 7th Workshop on Linked Data on the Web*, 2014.

[12] Pablo N Mendes, Max Jakob, Andrés García-Silva, and Christian Bizer. DBpedia Spotlight: shedding light on the Web of Documents. In *Proceedings of the 7th International Conference on Semantic Systems*, pages 1–8. ACM, 2011.

[13] Emmanuel Pietriga, Christian Bizer, David Karger, and Ryan Lee. Fresnel: A browser-independent presentation vocabulary for rdf. In Isabel Cruz, Stefan Decker, Dean Allemang, Chris Preist, Daniel Schwabe, Peter Mika, Mike Uschold, and LoraM. Aroyo, editors, *The Semantic Web - ISWC 2006*, volume 4273 of *Lecture Notes in Computer Science*, pages 158–171. Springer Berlin Heidelberg, 2006.

[14] Andreas Schultz, Andrea Matteini, Robert Isele, Pablo N Mendes, Christian Bizer, and Christian Becker. LDIF - A Framework for Large-Scale Linked Data Integration. In *21st International World Wide Web Conference (WWW 2012), Developers Track, Lyon, France*, 2012.

[15] Claus Stadler, Jens Lehmann, Konrad Höffner, and Sören Auer. Linkedgeodata: A core for a web of spatial open data. *Semantic Web Journal*, 3(4):333–354, 2012.

[16] Claus Stadler, Michael Martin, and Sören Auer. Exploring the Web of Spatial Data with Facete. In *Companion proceedings of 23rd International World Wide Web Conference (WWW)*, pages 175–178, 2014.

[17] Timo Stegemann, Jürgen Ziegler, Tim Hussein, and Werner Gaulke. Interactive construction of semantic widgets for visualizing Semantic Web Data. In *Proceedings of the 4th ACM SIGCHI symposium on Engineering interactive computing systems*, pages 157–162. ACM, 2012.

[18] Amrapali Zaveri, Dimitris Kontokostas, Mohamed A. Sherif, Lorenz Bühmann, Mohamed Morsey, Sören Auer, and Jens Lehmann. User-driven quality evaluation of dbpedia. In *To appear in Proceedings of 9th International Conference on Semantic Systems, I-SEMANTICS '13, Graz, Austria, September 4-6, 2013*, pages 97–104. ACM, 2013.