Published as http://dx.doi.org/10.1007/s00799-014-0130-2

Int J Digit Libr manuscript No.
(will be inserted by the editor)

Exploring publication metadata graphs with the LODmilla browser and editor

András Micsik · Sándor Turbucz · Zoltán Tóth

Received: date / Accepted: date

Abstract With the LODmilla browser we try to support Linked Data exploration in a generic way learning from the twenty years of web browser evolution as well as from scholars' opinions who try to use it as a research exploration tool. In this paper generic functions for LOD browsing are presented, and it is also explained what kind of information search tactics they enable with Linked Data describing publications. Furthermore, LODmilla also supports the sharing of graph views and the correction of LOD data during browsing.

Keywords Linked Open Data · Semantic Web · graph exploration · provenance

1 Introduction

The Semantic Web initiative provided the framework and the tools for sharing machine-understandable data on the Web. In 2006 Tim Berners-Lee outlined best practice for publishing and connecting structured data on the Web: the Linked Data principles [5]. This bases the identification and linking of semantic entities on so-called dereferenceable URIs, which can be used to retrieve more and meaningful information on the referenced object. On the other hand, Open Data refers to the open access to data in open (non-proprietary) formats. The merge of the two concepts became very popular in last years and was named as Linked Open Data (LOD) [7]. The LOD cloud diagram [9] recorded the growth of available LOD data until 2011, and counted 295 datasets in 2011 containing more than 31,000 million triples.

András Micsik · Sándor Turbucz · Zoltán Tóth
MTA SZTAKI, Budapest, Hungary
E-mail: {micsik,sandor.turbucz,zoltan.toth}@sztaki.mta.hu

Although the Semantic Web and LOD are meant for machine processable data, their use by humans cannot be avoided. Semantic data is often the only available place for the information sought, and furthermore it is usually more accurate and richer than any human-readable representation. Therefore, it is the task of the IT to provide nice and useful presentations of Semantic Data for humans. The problem with these visualizations is that they are often not generic, but ad-hoc; they are capable of presenting limited types of datasets only. While the World Wide Web had its generic visualization method, the web browser since the very beginning, the LOD cloud is still missing generic, user-friendly software tools for browsing, information search and exploration. In this paper we investigate previous generic LOD visualization approaches and present our own ideas in this respect which were implemented in the LODmilla prototype.

The next section of the paper gives an overview of approaches for browsing and visualization of Linked Data. Section 3 describes our suggestion for human LOD browsing, while Section 4 explains the server side architecture that supports the implemented visual graph queries. Section 5 introduces our publication data set which is used as a starting point for research activities exploration detailed in Section 6 before the conclusion.

2 Related work

Most approaches for presenting LOD for humans are dedicated to specific purpose and specific datasets, see for example [3]. The obvious solution for all-purpose LOD browsing is a pure text-based approach (e.g. Vir-

tuoso faceted browser[1] or Graphity[2]), where usually a single resource with all its referring triples are listed. In this case one can read data properties such as names, birthdate, etc., and also see connected resources as links. Clicking on a connected resource presents the same view for the selected new resource. The disadvantage of the pure textual approach is that we see one resource only and the graph structure of connections is not displayed. We are used to this in case of hypertext, but in RDF the connections convey elementary information, for which users should have an overview. Therefore, the combination of graphical and textual browsers is a more popular approach, and in the rest of the section we provide an overview of these.

Some people argue [12] that although an RDF dataset is a 'big fat graph', one does not always want to view them as graphs, because graph manipulation does not scale well and graph views are unnatural presentations for some purposes. On the other hand, the density, the clusters, the neighbourhood in a graph convey important, inductive information. Typicaly, when one zooms into a graph, the broader context gets lost. It is the task of the visualization to keep enough context around a focus in the graph. In LODmilla we provide selective expansion from a focus to solve this problem, so that users can quickly and flexibly decide on how much context they want to see.

There are various ideas for dynamic, free form GUIs for LOD, where a large variety of information seeking strategies can be combined. Tabulator [6], for example, fills cells with RDF data and expands them into bigger tables or converts it into a timeline or a map. mSWB is a Semantic Web Browser for mobile phones [14], where the limitations of small screens and touch-based navigation are added to the list of previously mentioned problems.

In [10] Dadzie and Rowe provide a rich survey of Linked Data visualisation approaches and also analyze some requirements for such applications. In our approach we try to address many of their requirements for example intuitive navigation through LD structures, data exploration to understand the structure of the dataset, advanced qerying.

One thing that differentiates the applications using the semantic web is the level where they handle the data. As [11] points out, the grouping by the granularity of information can be at collection level, resource level or intra-resource level.

While the collection level approach focuses on providing a general overview of a set of data, and mostly used for predictions, the resource level shows the attributes of the individual resources, and visualizes the connections between them, hence it provides more details on individual resources. Intra-resource level approaches show the distribution of the topics and attributes in a single resource, and they are used mostly for deeper analysis. In this work we aim at the resource level and try to point out the strong and weak points of related other work.

LodLive[3] represents the LOD resources and their connections in a graph structure. The visual design here is plain and simple, so it is relatively easy to understand the whole concept. Even so, the resources, represented by circles, do not contain enough information for the first sight. We only see a circle, with the resource labels listed in different languages, plus some other circles around it. If we would like to know more about a resource, we have to open its detail box on the right. Here we can see the data properties attached to the selected node in a pre-processed format, for example image URLs are detected and shown, geographical location is extracted and put on a map, etc. The major drawback of LodLive is the pure navigation on connections. Connections are grouped by property, and visualized as expanding small circles around the resource circle. This gives a limitation on the number of connections that can be shown, and in fact, LodLive truncates the shown connections to the first 30-40 for each connection type which results in information loss. It is also hard to see where the connection points to, as only the resource URIs are shown as a hover for each small circle. Resource URIs can be quite cryptic for humans when they contain numeric identifiers. Therefore, in LODmilla we aim at showing the labels or titles of connection endpoint resources. LodLive has a nice design, but quite often the usability is sacrificed on the design. As an advantage, it is made as a pure HTML5 browser, which can be run in any modern browser.

OOBIAN[4] is a feature-rich, well-designed and useful LOD browser implemented in Silverlight. Technically this is a drawback as Silverlight is not available in all browsers. OOBIAN consists of several views: a graph view, a textual reader, a file explorer and a map. This application combines the visual and text based approaches, but it cannot be used for advanced purposes. It is a good LOD browser to jump from one node to another or to filter properties, but there is a main limitation that one can see only a single resource and its connections in the graph view.

Microsoft Academic Search is a special tool for finding researchers, their publications, and the relations between these. It includes a Silverlight-based graph mod-

[1] http://dbpedia.org/fct/
[2] http://graphity.org
[3] http://en.lodlive.it/
[4] http://oobian.com/

ule named Visual Explorer[5] where one can visualize the connections between people, show the links between co-authors, and see the citation graphs for authors. The main problem with this approach is that it is limited to a given scenario and requires the internal database of Microsoft which is not open for any other organizations.

The VisualDataWeb[6] project produced a set of very interesting graphical user interfaces for the Semantic Web: RelFinder, gFacet, tFacet, SemLens . The RelFinder helps to find connection paths between selected resources. This is a very useful function if we want to know how two objects are related to each other. In LODmilla we implemented a similar function using a different solution in order to find paths including nodes from different RDF stores. gFacet and tFacet are the graph-based and textual implementations of faceted browsing of RDF data. SemLens provides tables and plots to analyse trends and correlations in RDF data. These tools cover specific needs for RDF data consumers, and may be applied as add-ons in future generic LOD browsers.

3 A generic LOD browser

Browsing the web is a commodity today with a number of web browsers on the market. These browsers share some default, fundamental controls and functions, which provide the basic browsing experience, as a result of crystallization during the last two decades. With any web browser we can open URLs, follow links, search for text on the page, save a copy of the page, etc.

Although Linked Data has a much shorter existence than WWW, we cannot find a tool that provides comfortable and visual browsing of semantic data and LOD resources. The tools we examined are either built for specific datasets or they are difficult to use and lack important visualization features. We believe that there is a need for generic LOD browsers, with a set of common basic features the users can learn and get used to. This would greatly increase the impact and usefulness of LOD.

The following basic actions for generic LOD browsers can be identified:

- Visual representation of multiple resources and properties (most probably as a graph),
- Opening resources, viewing object and data properties,
- Searching in the graph,
- Managing selections,
- Saving current view,
- Sharing views with others,
- Undoing previous actions (as a replacement of web browser history).

We implemented LODmilla as our prototype solution for the goals listed above. LODmilla is a graph based browser, running in conventional web browsers, developed using HTML, CSS and Javascript. While it is primarily visual, it also contains textual representations of resource properties in order to com-bine the best of both worlds. Its goal is to provide a simple, yet feature-rich application for the interactive exploration of LOD content residing in multiple knowledge bases. By its design, LODmilla does not hide any information available in RDF from the users, but it tries to organize and pre-process presented data. For example, incoming and outgoing properties are grouped by property type, and for the data properties URLs are made clickable, image URLs are shown inline and geographic locations are shown on a map. The work in the field of user experience is still in progress, but our long-term approach is to extend the interface with more advanced operations in a palette-like fashion, which work similarly to usual image manipulation software (e.g. Gimp).

3.1 Frontend

The goal was to implement a solution for a wide set of browsers. Therefore, HTML+CSS and SVG were selected as basic technologies, and two Javascript libraries as graphical toolkit: jsPlumb[7] and jQuery[8] . JsPlumb uses pure HTML+CSS for drawing the graph nodes, and SVG for drawing the links between them.

Figure **??** shows the four main parts of the web application:

- canvas
- palettes (top left)
- toolbar (bottom left)
- node inspector (right)

The canvas is the background, on which the graph structure is drawn. The palettes contain various actions, grouped as accordion styled menu items. We have a toolbar at the bottom, with the standard operations like load, save, etc. The triples referring to a node (LOD resource) are listed in the inspector window on the right, which opens by clicking the I (information) button in any node. The inspector window - unlike the other three main elements - can be moved, resized, and closed.

Complex operations can be started from the palettes, while simple ones can be found in every node. Each

[5] http://academic.research.microsoft.com/VisualExplorer
[6] http://www.visualdataweb.org/
[7] http://jsplumbtoolkit.com/
[8] http://jquery.com/

node has the following basic information in it: its datastore, its title or label, and the number of data properties and object properties associated with the resource. Additionally, an image representation (if found) or an icon based on the resource type is put in the middle of the node box.

Nodes may have the following actions: remove from the canvas, open details in the inspector window and select/highlight (with the star icon). Furthermore nodes can be moved around, and we can zoom or pan the whole graph view. Actions in the toolbar affect the whole view, while palette actions are used to manipulate a set of (highlighted) nodes or to display new nodes.

Opening resources

The first step when using the LOD browser is to open some nodes. This can be done by pasting a resource URI, but for some datasets we offer autocomplete search as well: by typing part of the resource label, one can choose from offered resources. The information associated with the resource can be browsed in the inspector window, and some new nodes can be opened by clicking on selected connections in the list.

Searching in open resources

Resources may have a lot of properties and long texts, which one may not want to read through. One palette item serves for searching text patterns in the content of all shown graph nodes. The search results appear as dynamic autocomplete suggestions under the search input box.

Searching for new resources

When we do not find the requested information in open nodes, we can try to expand our graph view with new nodes. This is also a search function; it finds the resources which somehow contain the given query word(s) in associated triples, and are connected to a selected resource. Figure 1 shows the use case of searching the word semantic from the starting node representing one of the authors in the middle. All the surrounding document resources are the search results containing the word semantic in their content.

Saving and sharing

The graph of shown RDF nodes and connections may demonstrate a new finding, or record a certain state of knowledge, which may be useful in the future for the user who created it. We offer the ability to fully save a graph state under user-given title, and load it later into the browser. Saved graphs can also be shared via a unique URL.

Undo

As some actions may unexpectedly cover the canvas with many new nodes, an Undo function has also been implemented, which reverts the last action the user made.

Selecting nodes

All nodes on the canvas can be marked one-by-one or in groups. One method to achieve this is clicking on the star in the top-left corner of the nodes. Another method is using the 'Select nodes' palette on the left, which can mark all nodes or nodes of the same type. Multiple node selection is also possible with the mouse as in most drawing software. Marking nodes provides the starting point for operations on node groups such as search. For example, we can select the person type resources, and perform a search for a project name in those nodes, this way finding the ones somehow related to the project.

3.2 Editing mode

The need for authoring Linked Data in a browser was also identified in [10]. It is more natural to draw connections in a graph than adding triples to a triple store. Therefore we extended LODmilla with an editing mode, and provide operations for changing the graph during exploration. This can be typically used to correct errors and to complete missing parts of information.

Although there is the SPARQL Update language and protocol [18] to inject changes into a triple store, it is not widely used yet, probably because of the authorization issues. Unlike editing Wikis, the joint editing of triples in a datastore is not prevalent today. In LODmilla, deletions of triples may affect several datasets, and it also requires a decision in which dataset to store triple insertions. The typical user cannot be expected to make such decisions. In order to avoid such problems and to reach a simple yet testable solution, we decided to collect user modifications in a local edit buffer.

The edit buffer contains triples with a timestamp and a flag indicating insertion or deletion. The content of the buffer can be retrieved as two triple sets: deleted triples and inserted triples. These triple sets can be sent to the datastore as a SPARQL Update, or they can be sent to an administrator who is authorized to perform these changes on the datasets. The edit mode is an aid to laymen for correcting or extending datasets without precise knowledge of RDF and other related specifications.

Users may insert and delete nodes or connections, drag connection endpoints in the graph. Furthermore, some operations can be performed in the infobox as well: connections may be deleted, data properties may be added or deleted. At any time, the modifications can be copied to the clipboard in Turtle format. Further plans include to save the modification list on the backend, and to enable users to have joint editing sessions via shared graph views. Figure 2 shows the edit

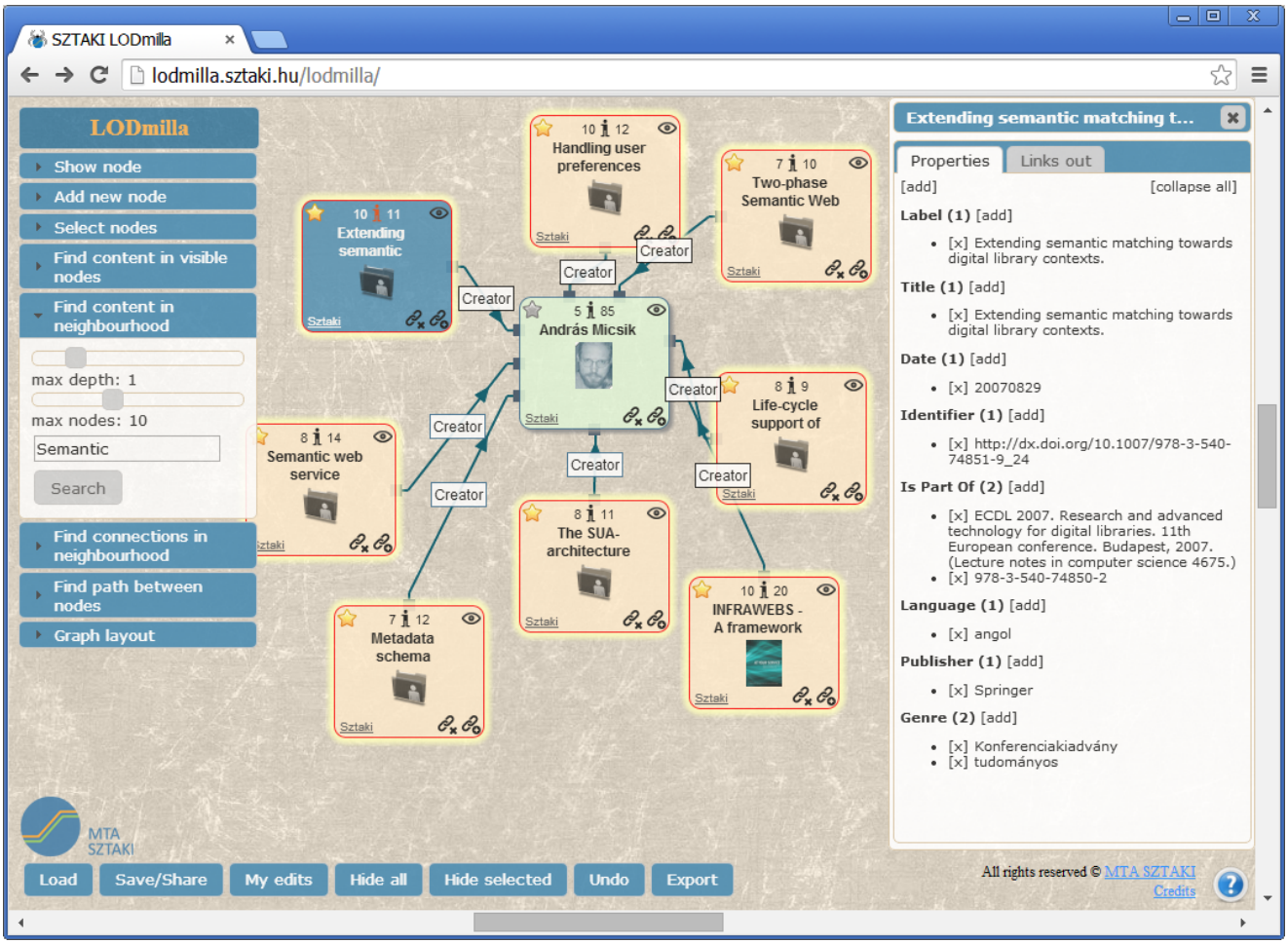

Fig. 1 Expanding the graph via text search

buffer displayed after the user pushed the "My edits" button.

While editing, user actions has to be translated into triple insertions and deletions. For example, moving a connection end from node A to node B generates one triple deletion and one triple insertion. While the translation of operations on graph edges to RDF operations is straightforward, the operations on nodes raises some issues. Node deletions may have several meanings from the RDF viewpoint as a node does not exist per se in an RDF dataset. A node can be deleted by deleting all triples referring to this URI either as subject or object. But one can never be sure to find all such triples from all datasets over the world. Another interpretation of node deletion may be to delete triples where the node is the subject in the current dataset. Our intention is not to hide the RDF nature of the graph, but to reflect it by the implementation of the editing actions. Therefore, we have an icon for simply hiding the node, and another to delete all connections of the node. This latter operation deletes the currently visible connections only, and leaves hidden connections intact.

Regarding node insertion, we ask for the label and type of the new node, and in order to avoid URI collisions, the URI is generated as a globally unique id with the prefix selected by the user. The new node can then be displayed on the screen using an optional thumbnail URL to show as an icon. As a result, new nodes manifest as two or three new RDF triples.

Finally, in the inspector each property-value pair represents one triple, so the change, insert or delete operations have natural meanings in this case. It will require more time to find all discrepancies in graph-based editing of RDF, and to reach a common set of translations from graph operations to RDF modifications.

4 Backend

Most of the browsing functionality does not rely on a server, and thus our tool could work without a ded-

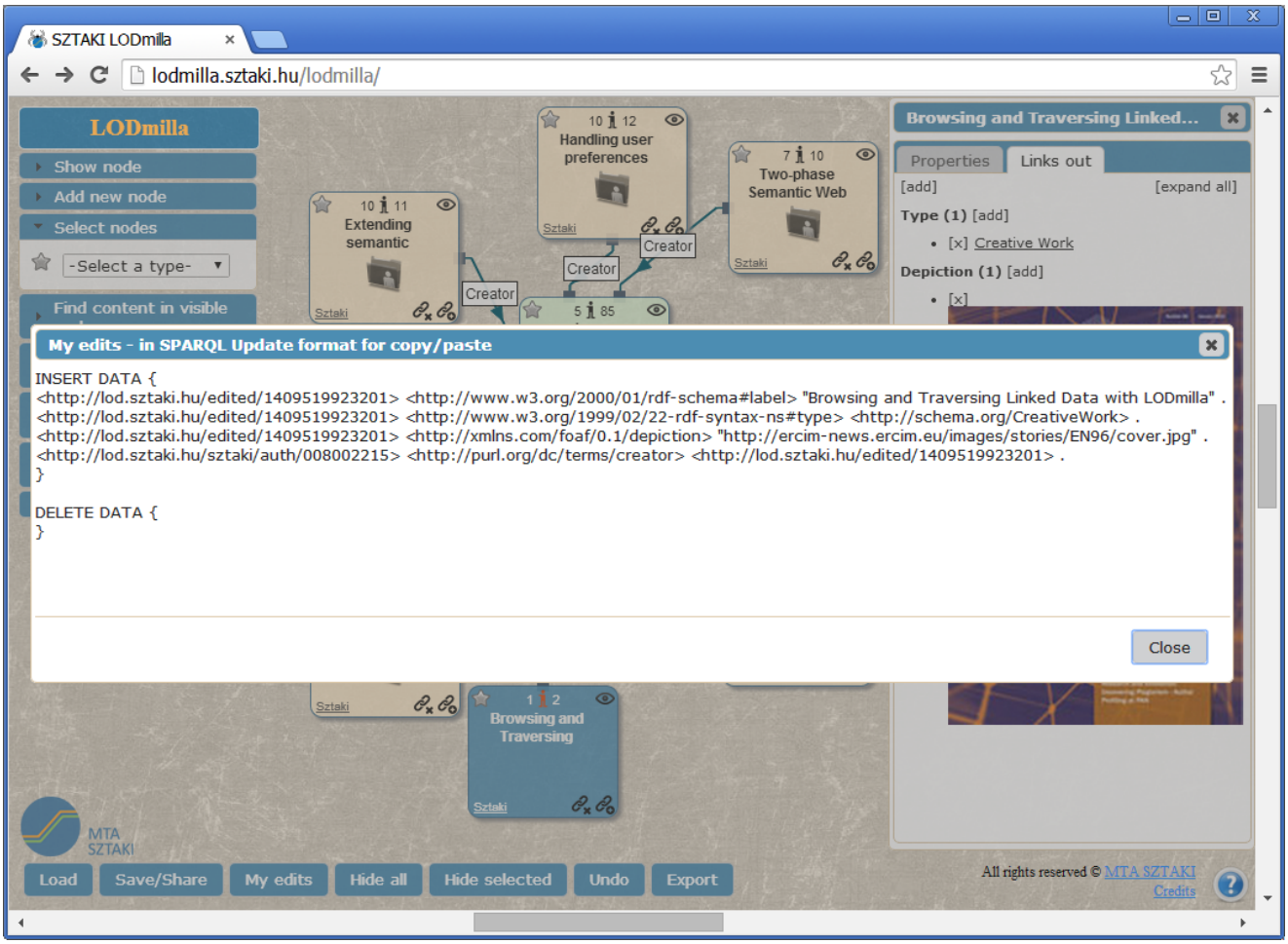

Fig. 2 The edit buffer containing changes

icated server. In order to support saving and sharing users views, we had to implement a server side component as well. The backend has an additional benefit for performance as it may load information faster and cache visited nodes. Finally, we moved most of the search operations to the backend because of these reasons. Search operations use both graph traversal and SPARQL queries, as we cannot always find an open SPARQL endpoint for datasets.

We put a requirement that our solution should work on as many datasets as possible, and it should use the latest information available, so harvesting and preprocessing datasets as in [19] was not a viable option in our case. These presumptions lead us to a graph traversal which can use either a SPARQL query (DESCRIBE for example), or dereferenceable URIs to fetch the connections of resources. Incoming RDF can be parsed as Turtle, RDF/XML, JSON, etc. in the backend using the Jena toolkit [15]. Three variations of LOD graph search have been implemented:

- Content search: we are searching neighbour nodes with data properties containing the given search pattern,
- Connection search: we are searching neighbour nodes for object property names matching the search patterns,
- Path finder: paths are sought between selected nodes.

In all cases we wanted to avoid solutions that work in single datasets only and solutions which use preprocessing of whole datasets. These requirements lead to several problems: first, the quality of the RDF stores is quite different in capabilities, availability and speed, which has big impact on the performance and quality of the graph traversing process. Some of the RDF nodes might not be available during the search process, or they can be slowly harvested. The second problem is that the world-wide LOD graph is huge: nodes may have 500 or 1000 connections, and a 2-step path may cover 3 different RDF stores. The third problem also comes from the heterogeneity of our data sources: links

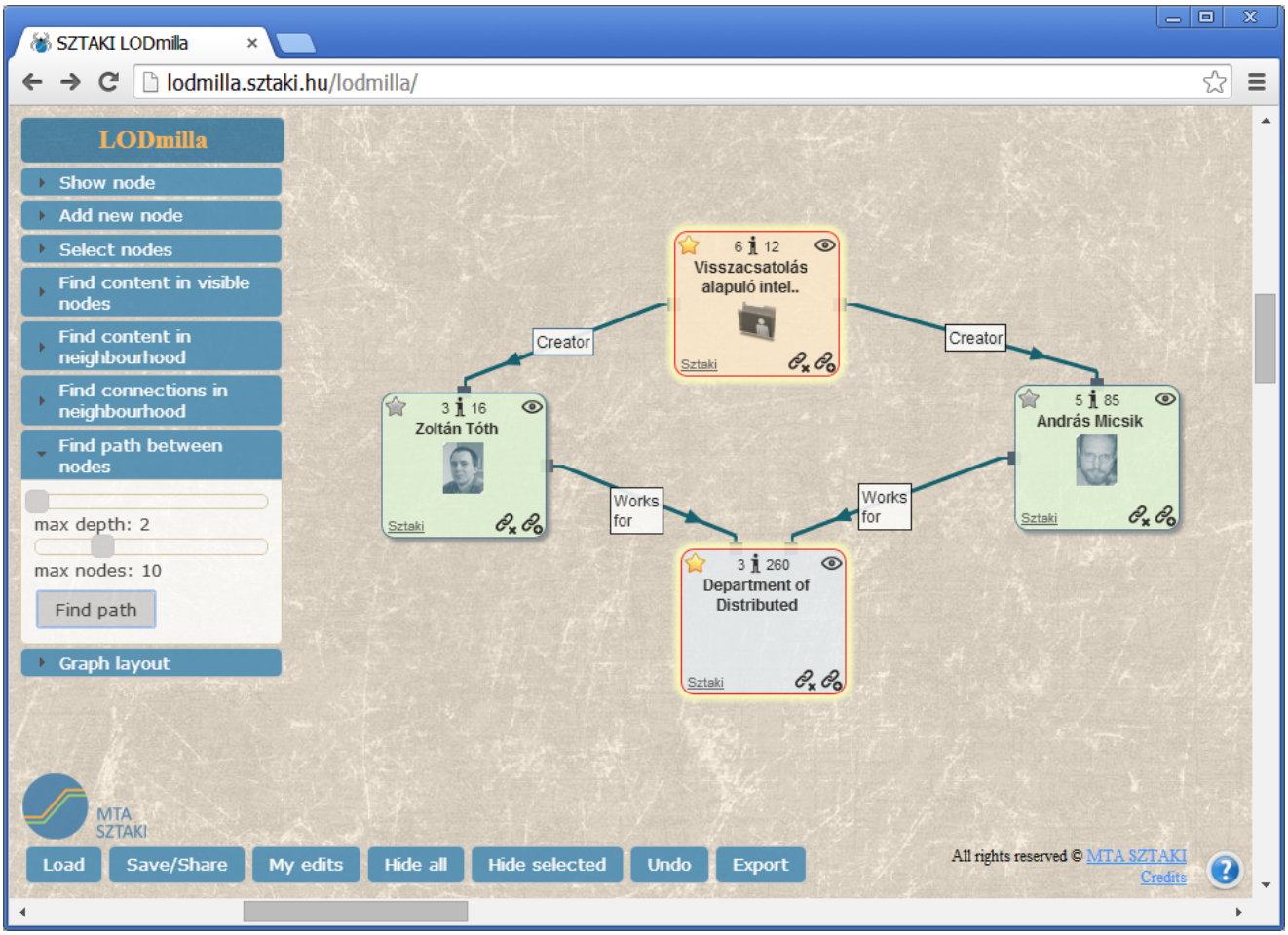

Fig. 3 Finding paths between two resources

between graph nodes sitting at different RDF stores are known only by one of the nodes (i.e. incoming links are not stored).

Because of these limitations, we chose to generate our graphs dynamically. When a user explores a part of the LOD graph, several search operations may be started in a sequence. These queries may be slow at first due to dynamic loading of nodes, but will get faster and faster after the graph area is cached.

As we only see a part of the whole graph at search time, most of the well-known fast graph search algorithms are not applicable in our case. We have to go back to A* style traversal and adapt it to our needs. It is hardly possible to estimate the distance to the goal node, but we can use some heuristics based on connection types. As a specificity of this task, there are paths we are simply not interested in, for example Book resources are all connected to the Book RDF class. Therefore, we simply do not follow a set of trivial links denoting type, language or format of nodes. The traversal of the remaining links may be ordered heuristically based on learning, this remains as future work in the project.

In the case of remote content search our task is to answer the question: does a node containing a given string in a data property exists in the neighbourhood of a given node? To answer this question breadth-first traversal in the RDF graph structure is applied. We have to limit this algorithm in several ways. First, a maximum depth is specified until the algorithm tries to find results. As the result set may easily grow to hundreds of nodes, which is unmanageable for the user, the number of result nodes is limited as well. The result of the search is shown as highlighted nodes in the graph for which the shortest paths from the start node are also displayed.

In the case of connection search only connection types matching the given search text(s) are followed. Multiple search items might be added divided by a separator. The traversal of links is also breadth-first. The result of the query is a set of nodes which are accessible via the matching connections. The relevant incoming

connections, just like in the case of the content search algorithm, might not be discovered, as these can only be retrieved using SPARQL endpoints, and even so, we cannot ask each SPARQL endpoint in the world for the existence of such connections.

For path finding the question is if a path exists between two RDF resources (Figure 3). Our implementation is similar to the Dijkstra algorithm with all edges having the same weight and where the graph is produced on the fly. The starting parameters of our implementation are the two starting nodes, the maximum depth of the search, and the maximum number of nodes we can handle as a response. In the first iteration a breadth-first traversing starts from both endpoints. Their connections are checked and if common nodes can be found, they will be accepted as results and the algorithm is finished. In each further iteration the nodes accessible from the already found nodes are checked. One such iteration means two steps in depth increment since we are growing our graph from both ends. The local target is to find nodes which have parents to both of the source nodes. When such node has been found, the resulting graph must be simplified. All nodes not on the common path are eliminated. In this case it still might happen that we have more nodes in the path than the number we can comfortably handle on the user interface.

5 Datasets used with LODmilla

The aim of LODmilla was to support zero configuration browsing of any LOD dataset. Therefore, we rely on dereferenceable URIs [17], and the ability to download resource descriptions in at least one of the popular Linked Data formats using content negotiation. The voiD vocabulary [2] provides a method for publishing high-level descriptions of datasets, including the main entry points such as the SPARQL endpoint. Although the use of voiD is still infrequent, we plan to rely on voiD descriptions to automatically configure the browser for the visited datasets.

Besides DBpedia, the most frequently used data for LODmilla is provided by the SZTAKI LOD service[9] containing two datasets: one about publication data and one with the contents of Hungarian archives based on the National Digital Data Archive of Hungary. The second dataset (11 million triples) contains information about books, movies, articles published in Hungary with links to other datasets such as DBpedia or VIAF.

The cultural dataset was produced from the results of OAI-PMH harvests from several Hungarian repositories. In this way we collected more than 800.000 Dublin Core records in XML, and converted them to RDF. The RDF conversion used DBpedia, schema.org, FOAF and dcterms schemas. We decided to use multiple type definitions in order to facilitate further processing of our data without reasoning under different schemas. As an example, person descriptions have the Person type from DBpedia, FOAF and schema.org schemas. The RDF result uses 17 dcterms properties to describe the cultural assets. The processing of creators took much extra effort. The original metadata records did not contain creator identifiers, and they were also quite dirty, for example birth dates and contribution role names were often put before or after the creator names. Therefore, we first applied some rules to detach dates and roles from names, and use them to enrich the description of persons and assets. Further then, we extracted cca. 375.000 creator occurences from the cultural metadata. These names we wanted to match with named authority records, or person records from VIAF, DBpedia and the Hungarian National Library. Hungarian names appear in different order: last name first. We also had to deal with the various ordering of name parts. Different name formats were recorded as dcterms alternative data property. The name linking process searched the three datasets for some name variations, and if there was an obvious match, a sameAs property was recorded for the author. In this way cca. 130.000 creators were linked to at least one other dataset.

A similar process was completed for the publication records of SZTAKI researchers. This resulted in a much smaller dataset, but it was much more interesting to our colleagues. The dataset contains the usual metadata such as title, publisher, date, subject and creators for each publication. Each publication is linked to the producing laboratory. In the future, the dataset will be extended with links to concepts representing keywords and topics of publications. The publication dataset was used to collect scenarios and opinions from researchers, discussed in the next section.

6 Using LODmilla for research activities exploration

We consulted twelve researchers in order to find out how they could use LODmilla for browsing publication metadata and what are their most frequent activities during exploration of such data. Among the researchers there were computer scientists, mathematicians, social scientists, a linguist and a Ph.D. student in philosophy. In the following we list some of their suggested exploration strategies and the possible tactics [4] to perform them using LODmilla.

[9] http://lod.sztaki.hu

Most of the collected exploration activities start from a small set of papers or authors. The elements of the set can be found in LODmilla by entering the keywords or person names into the node opener palette. It works with a SPARQL endpoint currently, or it can work using a global Semantic Web search engine such as Falcons[10]. These starting points represent interesting papers or researchers working on a field that meets our current interest.

From the starting points, the exploration tries to collect new information or interesting papers. It is very typical to traverse the links of creators and references, and thus surf on the links, which can also be done using Google Scholar (for example). The main difference is that LODmilla 'records' one's paths of surfing in the graph, and the visited nodes may be kept on the screen or dismissed if judged as irrelevant for the current search. The graph view also enables us to move faster back and forward on citation links between papers.

Quite easily one may litter the screen with many superfluous nodes. It is essential not to miss the goal and to keep the graph at a manageable size. One can speed up and focus exploration by surfing towards relevant directions, for example by searching for keywords in the neighbourhood of selected papers or authors using the palette for neighbourhood content search in LODmilla.

6.1 The author network

On the other hand, if one is interested in the author network, it can be expanded around selected authors using the Remote connection search palette in LODmilla. Discovering such connections is important in scholarly exploration: our interviewees were eager to know:

- if certain people have written papers together,
- if certain people have worked together at some location,
- if supervisors of Ph.D. students may know each other,
- if certain people could act as hidden influencers (this can be guessed in case of a common working place, for example)

Such kind of connections can be revealed with the help of the path finding palette of LODmilla. It is often important to find out that selected topics has been researched close to us. One can start a search for keywords in one's neighbourhood, and may find relevant papers by colleagues, or relevant theses done at a nearby department. Consequently, one can search for a given topic restricted to any institute or department, by starting the search from the node representing that unit.

[10] http://ws.nju.edu.cn/falcons/

6.2 The citation network

By opening the citation network, we see paper references as links between papers. This network needs to be filtered by date, because we are normally interested in recent evolution of research topics, so papers outside a period need to be hidden. This feature is not yet implemented in LODmilla. The citation network may reveal some key papers, which are cited from most other papers in the topic. Therefore, their indegree is the highest in the citation network. Highlighting such nodes is another new feature requirement for LODmilla. Such papers may be seen as the starting points of new research directions. Similar goal exists for authors as well; the authors with the most papers in the topic may be the 'grand masters' in that field.

Researchers are sometimes interested in circular citations, where authors regularly cite each other. It is good to know who are the members of such cliques, and whether one person is inside or outside. A graph representation is quite ideal for answering such questions.

Researchers seek for open access, and by using remote connection search in the graph, the nodes with connections for associated fulltext, research data or source code are highlighted.

6.3 Strategic maps

One of our interviewees aims at building 'strategic maps' for learning new fields, which initially contain the important papers and authors of the field, and then by understanding the relations among papers and the quality of papers, the most imortant papers are selected with minimal overlap in order to reduce reading time. For this a tactic he would like to apply is to find papers with the same authors in the same topic, which can be translated to multi-criteria path-finding. Another tactic needed is to rank paper nodes by their citedness, and to filter out less cited papers for example. The third requirement for the refinement of such maps to be able to show or hide nodes based on multiple criteria, for example paper nodes not containing certain keywords.

An interesting aspect here is the distance (or similarity) of papers, which can be represented by the length of the connecting edges traditionally. The similarity may be pre-calculated by some other service, but it can also be guessed using available node properties such as keyword lists. It is an interesting question how to represent similarity in LODmilla graphs.

Finding trends and state-of-the-art is the 'ultimate question' raised by the interviewees. These can be characterized by topological and statistical characteristics

of the graphs together. The papers setting the trend should be recent, but need to have relatively large number of citations.

Naturally, there are a bunch of other tasks in scholarly information retrieval which require a tabular or listing view, such as the statistics of publications and their types or citations. The traditional keyword search in the style of Google also remains indispensable.

6.4 Research data

Cyberscholarship is a rapidly expanding phenomenon, where new results are mined and discovered from the growing number of primary and secondary sources [13]. Murray-Rust suggests the continuous creation of semantic objects during research [16], which should be the basis of publication and could be an enabler to find undiscovered results. This means that besides the previously examined traditional publication data, the description of research data needs to serve an impotant role. The two kinds of descriptions are related and can be used together with the help of further improvement of data citation techniques. Although cyberscholarship typically involves text mining and data mining, there is an inevitable need to explore the results of mining, and browsers such as LODmilla can be used for that purpose in case of semantic data. LODmilla and similar software can be a part of the new service infrastructure required for cyberscholarship to support "self-organizing knowledge driven by human interaction" as reported by Larsen [13]. We are in the phase when semantic descriptions of primary data is becoming available, for example, myExperiment provides experiment descriptions in RDF [11]. In the area of social science the Data Documentation Initiative (DDI) is an international standard for the documentation of data. The DDI-RDF Discovery Vocabulary defines RDF description of social science research data in order to facilitate semantic searching [8]. When Linked Data about research data and publications will have rich interconnections, LODmilla may serve as a handy tool for the human exploration of such RDF datasets.

7 Conclusion

In this paper we argue that generic tools for exploring and navigating the LOD cloud are necessary not only for computers but for humans as well. The example of Web browsers show that functionality for using a similar technology converges to a common visualization and to a common set of functions. With the LODmilla browser we experiment and test these common functions for generic LOD browsing, with the aim to exploit the benefits of graph visualization, browseable lists and flexible search functions. It is also important to offer this functionality on all datasets matching minimal criteria (dereferenceable URIs or a SPARQL endpoint) without the need of manual configuration for each dataset.

Additionally, new ways of searching and exploring the LOD graph are needed. LODmilla supports several candidates for these: content search starting from a resource, finding paths between resources, expanding the graph via given connection types, etc. With this approach LODmilla is capable to handle, and more importantly connect most LOD knowledge bases easily and transparently, and provide a shared knowledge exploration and visualization experience for its users.

The features of the browser enable the users to access, visualize and explore all contextual and relational information of LOD resources including research datasets or publications as a particular application area. Most of the information seeking strategies suggested by scholars could be translated to some concrete tactics using the LODmilla browser, and we also received interesting new ideas to improve the exploration support of the browser.

LODmilla is an open web application[12], with its source code published on GitHub[13].

[11] http://rdf.myexperiment.org/

[12] Accessible at http://lodmilla.sztaki.hu/

[13] https://github.com/dsd-sztaki-hu/LODmilla-frontend

References

1. Visual data web: visually experiencing the data web. http://www.visualdataweb.org/, 2011.
2. Alexander, K., Cyganiak, R., Hausenblas, M., and Zhao, J. Describing Linked Datasets - On the Design and Usage of voiD, the 'Vocabulary of Interlinked Datasets'. In *WWW 2009 Workshop: Linked Data on the Web (LDOW2009)* (Madrid, Spain, 2009).
3. Badger, E. The best open data releases of 2012. http://www.theatlanticcities.com/technology/2012/12/best-open-data-releases-2012/4200/, 2012.
4. Bates, M. J. Information search tactics. *Journal of the American Society for Information Science 30*, 4 (1979), 205–214.
5. Berners-Lee, T. Linked-data design issues. W3C design issue document, June 2009. http://www.w3.org/DesignIssue/LinkedData.html.
6. Berners-Lee, T., Chen, Y., Chilton, L., Connolly, D., Dhanaraj, R., Hollenbach, J., Lerer, A., and Sheets, D. Tabulator: Exploring and analyzing linked data on the semantic web. In *In Procedings of the 3rd International Semantic Web User Interaction Workshop (SWUI06)* (2006).

7. Bizer, C. The emerging web of linked data. *IEEE Intelligent Systems 24*, 5 (2009), 87–92.
8. Bosch, T., Cyganiak, R., Gregory, A., and Wackerow, J. Ddi-rdf discovery vocabulary: A metadata vocabulary for documenting research and survey data. In *LDOW* (2013), C. Bizer, T. Heath, T. Berners-Lee, M. Hausenblas, and S. Auer, Eds., vol. 996 of *CEUR Workshop Proceedings*, CEUR-WS.org.
9. Cyganiak, R., and Jentzsch, A. The linking open data cloud diagram. http://lod-cloud.net/, Sept. 2011.
10. Dadzie, A.-S., and Rowe, M. Approaches to visualising linked data: A survey. *Semantic Web 2*, 2 (2011), 89–124.
11. Herrmannova, D., and Knoth, P. Visual search for supporting content exploration in large document collections. *D-Lib Magazine 18*, 7/8 (2012).
12. Karger, D., and Schraefel, M. The pathetic fallacy of rdf. Position Paper for SWUI06, 2006.
13. Larsen, R. L. On the threshold of cyberscholarship. *The Journal of Elenctronic Publishing 11*, 1 (2008).
14. Matuszka, T., Gombos, G., and Kiss, A. mswb: Towards a mobile semantic web browser. In *Mobile Web Information Systems*, I. Awan, M. Younas, X. Franch, and C. Quer, Eds., vol. 8640 of *Lecture Notes in Computer Science*. Springer International Publishing, 2014, pp. 165–175.
15. McBride, B. Jena: a semantic web toolkit. *IEEE Internet Computing 6*, 6 (2002), 55–59.
16. Murray-Rust, P. Semantic science and its communication - a personal view. *J. Cheminformatics 3* (2011), 48.
17. Sauermann, L., and Cyganiak, R. *Cool URIs for the Semantic Web*. W3C Interest Group Note, W3C, December 2008.
18. Schenk, S., Gearon, P., and Passant, A. Sparql 1.1 update. Tech. rep., W3C, 2008. Published online on October 14th, 2010 at `http://www.w3.org/TR/2010/WD-sparql11-update-20101014/`.
19. Vocht, L. D., Coppens, S., Verborgh, R., Sande, M. V., Mannens, E., and de Walle, R. V. Discovering meaningful connections between resources in the web of data. In *LDOW* (2013), C. Bizer, T. Heath, T. Berners-Lee, M. Hausenblas, and S. Auer, Eds., vol. 996 of *CEUR Workshop Proceedings*, CEUR-WS.org.