Adapting the LodView RDF Browser for Navigation over the Multilingual Linguistic Linked Open Data Cloud

Alexander Kirillovich
Kazan Federal University,
Joint Supercomputer Center
of the Russian Academy of Sciences
Kazan, Russia
alik.kirillovich@gmail.com

Konstantin Nikolaev
Kazan Federal University,
Joint Supercomputer Center
of the Russian Academy of Sciences
Kazan, Russia
konnikolaeff@yandex.ru

Abstract—The paper is dedicated to the use of LodView for navigation over the multilingual Linguistic Linked Open Data cloud. First, we define the class of Pubby-like tools, that LodView belongs to, and clarify the relation of this class to the classes of URI dereferenciation tools, RDF browsers and LOD visualization tools. Second, we reveal several limitations of LodView that impede its use for the designated purpose, and propose improvements to be made for fixing these limitations. These improvements are: 1) resolution of Cyrillic URIs; 2) decoding Cyrillic URIs in Turtle representations of resources; 3) support of Cyrillic literals; 4) user-friendly URLs for RDF representations of resources; 5) support of hash URIs; 6) expanding nested resources; 7) support of RDF collections; 8) pagination of resource property values and; 9) support of \( \LaTeX \) math notation. Third, we partially implement several of the proposed improvements.

Index Terms—LodView, Pubby-class tool, RDF browser, dereferenciation tool, Linked Data visualization, Linguistic LOD

I. INTRODUCTION

Digital language resources are key to linguistics research and natural language processing. Among these resources are corpora, thesauri, lexicons, grammatical dictionaries, valence dictionaries, linguistic data categories registers, and other.

Publishing linguistic resources according to the principles of Linked Open Data (LOD) [1]–[3] is a recently established trend, that has led to the emergence of the fast-growing Linguistic Linked Open Data (LLOD) cloud [4], [5]. It offers a number of advantages, including structural and conceptual interoperability, and integration of heterogeneous linguistic resources to solve a common problem.

As a part of the global LLOD project we are working on developing the fragment of the LLOD cloud for Russian and minority languages of Russia. This fragment integrates several resources such as the RuThes [6], TatThes [7], TatWordNet [8], TatVerbBank, the Russian corpus “OpenCorpora” [9], the Tatar national corpus “Tugan Tel” [10]–[12], the parallel informal/formal corpus of educational mathematical texts in Russian [13], the linguistic layer of the educational mathematical ontology OnToMathEDU [14]–[16] and other.

In accordance with Linked Open Data best practices, the integrated resources have to be available via RDF dumps, SPARQL endpoints and dereferenceable URIs. For navigation over the LOD cloud via dereferenceable URIs, a specialized tool is required. One of the most popular navigation tools is the LodView RDF browser.

LodView (https://github.com/dvcama/LodView) is a web application for dereferencing URIs of the RDF resources stored in the SPARQL-exposed datasets. When a human user requests the URI of an RDF resource, LodView retrieves information of the resource in the dataset and visualizes this information as a human-readable HTML page. On the page, the resource properties are represented in a tabular form. Each row represents a property-value pair; the first cell of the row contains the property name and the second contains the property value or values. If a property value is the URI of another resource,
this value is rendered as a link. A user may follow the link and get representation of the another resource, thus navigating over RDF dataset. The HTML page also contains links to the RDF documents with machine-readable representation of the resource (in different serialization format, including N-Triples, Turtle, RDF/XML and JSON-LD). When the URI is requested not by human but by a software agent, LodView immediately returns the machine-readable RDF representation.

Fig. 1 shows the place of LodView among other LOD applications.

Fig. 2 represents an example how LodView visualizes the ‘автомобиль’ (‘automobile’) lexical entry.

However, LodView has several limitations that impede its use for navigation over the multilingual LLOD cloud. In this paper, we continue the work initiated in [17], [18] and present the first version of our LodView fork, adapted to address these limitations.

The rest of the paper is organized as following. In Section 2 we define the class of Pubby-like tools that LodView belongs to and outline the relationships of this class with other classes of LOD applications. In Section 3 we list the limitations of LodViews and propose improvements to be made for fixing these limitations. Section 4 describes the improvements, that have already been made by us. In Conclusion, we outline the directions of future work.

II. RELATED WORKS

In this section we clarify the place of LodView among other LOD applications.

LodView belongs to a class of tools, that we called the Pubby class after its first prominent representative. We define a Pubby-class tool as a LOD application which is:

1) a URI dereferenciation tool, i.e. when a client requests the URI of an RDF resource via HTTP protocol, the tool handles the request and responds with a resource representation;

2) a LOD visualization tool, i.e. it represents the resource in human-friendly visual form;

3) a Linked Data or RDF browser, i.e. it allows a user to navigate between RDF resources of an RDF dataset or the entire LOD cloud, by following links in resource representations;

4) and a SPARQL endpoint interface, i.e. it retrieves information about resources from a SPARQL-exposed dataset.

Conceptually, these four classes are independent and so it is possible for example to conceive a URI dereferenciation tool that doesn’t allow a user to follow links or even doesn’t visualize a resource at all, dereferencing it just as a source RDF file in some textual serialization format. However, in practice,
every known URI dereferenciation tool is an RDF browser, and every known RDF browser is a LOD visualization tool (see the relationships between the classes at Fig. 1).

**Pubby-class tools.** Besides LodView, the Pubby class includes the following tools:

- Pubby A Java web application. Was used for publishing a lot of LOD datasets, including DBpedia. However, it hadn’t been updated in more than 10 years and now its GitHub repository is officially archived.
- LODDV A Java web application. Is used for publishing the AGROVOC Linked Dataset.
- Zazuko Trifid A Node.js-based web application. Is used in two open data platforms of Swiss Confederation.
- RDFBrowser A PHP web application. Is used by the OpenBudgets.eu platform.
- LOD A Node.js-based web application. Is used by two Spanish open data portals.

**Other Linked Data and RDF browsers.** While almost all Pubby-class tools are Linked Data browsers, there are Linked Data browsers that are not Pubby-class tools. Among the first such Linked Data browsers the well-known ones are Haystack [19], Discor and Tabulator [20]. Examples of the more recent developments are Phuzzy.link [21] and RDFSurveyor [22].

Besides Linked Data browsers, there are many LOD visualization tools of several other types, including graph-based and ontology visualization tools.

**Graph-based visualization tools.** While RDF browsers mainly work on the level of single RDF resources, graph-based tools work on the level of entire RDF graphs. One of the earliest tools in this category is IsaViz [16], a desktop application for authoring and visualizing RDF models, represented as relatively small local RDF files. In contrast, graphVizdb [23] enables the user to navigate over very-large RDF graphs. LodLive [24] and LODmilla [25, 26] provide incremental graph visualization. RelFinder [27] and WiSP [28] are intended to identify and visualize the paths between two selected RDF resources.

**Ontology visualization tools.** Examples of these tools are OWLPropViz, WebVOWL [29] and OWLGrEd [30].

For a book length overview of LOD visualization tools, we refer the interested reader to [31].

### III. Required improvements of LodView

In this section, we list the limitations of LodViews and propose improvements to be made to fix these limitations.

![404](http://localhost:5820/ru-thes-cloud/query
doesn’t contain any information about\n<http://lod.ruthes.org/resource/entry/RU-Д машина-n>]

**Figure 3.** Error occurred while resolving a URI that contains Cyrillic characters.

1) **Resolution of Cyrillic URIs.** In a LLOD dataset, the URI of an RDF resource may contain Cyrillic characters. For example, in the RuThes Cloud dataset, the URI of the lexical entry ‘машина’ (‘car’) is <http://lod.ruthes.org/resource/entry/RU-машина-n>. However, in some environments, resolving Cyrillic URIs by LodView leads to an error (see Fig. 3). LodView must resolve Cyrillic URIs in all environments.

2) **Decoding Cyrillic URIs in Turtle representations of resources.** In a machine-readable RDF representation of a resource in the Turtle serialization format, Cyrillic URIs are presented in an encoded form. Fig. 4A shows a fragment of the Turtle representation of the Automobil concept, that contains encoded URIs. Encoded URIs hinder readability of Turtle documents. In the Turtle representation of a resource, URIs must be presented in a decoded form (for example, as in Fig. 4B).

![LodView](https://cdn-images-1.medium.com/max/800/1*E4IzHxkRCAWCp5GrG8C2jA.png)

**Figur 4.** Fragment of a Turtle representation of a resource with encoded and decoded URIs.

#### A) Encoded URIs:

```
<resource/concept/154>
  a skos:Concept;
  ontolex:lexicalizedSense
    <resource/sense/154-машина-n-0>,
    <resource/sense/154-автомобиль-n-0>,
    ...>
```

#### B) Decoded URIs:

```
<resource/concept/154>
  a skos:Concept;
  ontolex:lexicalizedSense
    <resource/sense/154-машина-n>,
    <resource/sense/154-автомобиль-n>,
    ...>
```
3) Support of Cyrillic literals. In a machine-readable RDF representation of a resource, Cyrillic literals are in broken encoding. This happens even if the dataset was encoded by Unicode. Fig. 5A shows a Turtle representation of the RuThes Cloud resource, that contains literals in broken encoding. LodView must represent Cyrillic literals correctly (as in Fig. 5B).

A) Broken encoding (so-called mojibake):

```
<http://lod.ruthes.org/resource/ruthes>
a skos:ConceptScheme;
  rdfs:label
    "RuThes Cloud LLOD dataset"@en,
    "Набор лингвистических открытых связанных данных РуТез Cloud"@ru.
```

B) Recovered encoding:

```
<http://lod.ruthes.org/resource/ruthes>
a skos:ConceptScheme;
  rdfs:label
    "Russian Thesaurus Cloud LLOD dataset"@en,
    "Набор лингвистических открытых связанных данных РуТез Cloud"@ru.
```

Figure 5. Fragment of a Turtle representation of a resource with broken and recovered literals.

4) User-friendly URLs for RDF representations of resources. LodView resolves URI’s of RDF resources in accordance with the content negotiation principle. When the URI of an RDF resource is requested on by a human user, LodView redirects the user to the URI of the HTML page with human-readable resource representation. When the URI is requested by a software agent, LodView redirects the agent to the URL of RDF document with the machine-readable representation.

URLs of HTML pages are of user-friendly format: <RESOURCE_URI.html>. For example, the resource identified by the URI <http://lod.ruthes.org/resource/entry/RU-машинa-n> is represented by the HTML document located at the URL <http://lod.ruthes.org/resource/entry/RU-машинa-n.html>. However, URLs of RDF documents are rather cumbersome. For example, the Turtle representation of the aforementioned resource is located at the URI <http://lod.ruthes.org/resource/entry/RU-машинa-n?output=appli-cation%2Frdf%2Bxml>. In order the URL’s of RDF documents can be easy to read, they must be of user-friendly formats: <RESOURCE_URI.ttl>, <RESOURCE_URI.n3>, etc. For example, the Turtle representation of the <http://lod.ruthes.org/resource/entry/RU-машинa-n> resource must be assigned the URI <http://lod.ruthes.org/resource/entry/RU-машинa-n.ttl>.

5) Support of hash URIs. In LLOD datasets, RDF resources may be identified by hash URIs. A hash URI is a URI, that contains a special part separated from the base part of the URI by a hash symbol (#). This special part is called the ‘fragment identifier’. Usually, a hash URI identify a resource, that is somehow subordinated to the resource identified by the base part of this URI. For example, in the Princeton WordNet dataset, the hash URI <http://wordnet-rdf.rutgers.edu/wn31/cat-n#CanonicalForm> identifies the canonical form of the lexical entry ‘cat’, while the lexical entry itself is identified by the base part <http://wordnet-rdf.rutgers.edu/wn31/cat-n>. In this case, the URI of the canonical form of the lexical unit is formed by the URI of this lexical entry concatenated with the #CanonicalForm fragment identifier.

However, LodView can’t resolve URIs with fragment identifiers. So, for example, when a user try to retrieve the hash URI <http://wordnet-rdf.rutgers.edu/wn31/cat-n#CanonicalForm>, LodView sends a representation not of the resource identified by this URI, but of the resource identified by the URI <http://wordnet-rdf.rutgers.edu/wn31/cat-n>. This problem is caused by the HTTP protocol: when a client wants to retrieve a hash URI, the HTTP protocol requires the fragment part to be stripped off before requesting the URI from the server. This problem can be solved as follows: when LodView resolves a URI, its response must contain not only information of the requested resource, but also of all other resources identified by hash URIs with the same base part. For example, the representation of the resource identified by <http://wordnet-rdf.rutgers.edu/wn31/cat-n> must contain information of this resource along with information of the resource identified by the hash URI <http://wordnet-rdf.rutgers.edu/wn31/cat-n#CanonicalForm>.

6) Expanding nested resources. In LodView, the HTML representation of an RDF resource contains the table with the resource properties. In each row, the first cell contains the property name, and the second cell contains its value. The value of a property may be the URI of another, nested, resource. For the user, the only way to retrieve information about this nested resource is to follow the link and go to another HTML page. This need to leave the current page may spoil the user experience. LodView must allow the user...
to expand nested resources enabling him or her to see the properties of these resources without leaving the current page.

When the property value is a blank node, this value is represented in the table by the surrogate blank node ID. The description of this blank node itself is located at the bottom of the page. See for example Fig. 6, where a blank node is a value of the wn:definition property. In order to see information of the blank node, a user have to scroll the page. Again, LodView must allow the user to expand nested blank nodes to see its properties in place without scrolling the page.

7) Support of RDF collections. LLOD datasets may contain RDF collections. According to the RDF standard, a collection is represented as a low-level list structure in the RDF graph. When LodView visualize an RDF collection, it visualizes not the collection itself, but the underlying list structure. This way of visualizing makes comprehension of the collection difficult. An example of visualization of the RDF list < #comp-объект>, #comp-культурного>, <#comp-наследия > is represented at Fig 7. LodView must visualize RDF collections as a high-level sequence of its members.

8) Pagination of resource property values. An RDF resource may have an arbitrary number of values for a particular property. And this number can be very big. For example, the RuWordNet thesaurus is linked by the lime:entry property with more than 100 thousand lexical entries. However, LodView tries to display all these values at once, which can overwhelm the user with the amount of data, significantly slow down page loading and even cause server overload. To prevent these negative consequences, LodView must limit the number of the displayed property values by providing a pagination widget or just a “Load more” button.

9) Support of LATEX math notation. LOD datasets may contain literals with math notation in the LATEX format. LodView doesn’t render this notation and visualizes it just as its source LATEX code (see Fig. 8). LodView must be able to visualize math notation it the rendered form.

IV. IMPLEMENTATION OF THE IMPROVEMENTS

Currently, we have implemented the improvements ##1–4 and #9, and partially the improvements #5 and #6.

Issue #1 (Resolution of Cyrillic URIs). After analyzing the source code of LodView, we found that this problem is related to the fact that the URI text is represented in the SPARQLEndPoint class in Western ISO-8859-1 encoding. This problem was solved by adding a single line of code to the specified class that recodes the URIs from ISO-8859-1 to Unicode:

IRI = new String (URI.getBytes("ISO-8859-1"), "UTF-8");

After we solved this problem, LodView began to correctly resolve Cyrillic URIs and generate a web page displaying the corresponding resource when accessing them. However, after this, we found a new problem: the resource URI is also displayed incorrectly on the returned web page.

We solved this new problem by performing a similar recoding in the code of the corresponding web page (resource.jsp).

Issue #3 (Support of Cyrillic literals). This problem was solved in the same way as the previous one, using a similar recoding.

Issue #2 (Decoding Cyrillic URIs in Turtle representations of resources). To solve this problem, we have made changes to the ResourceBuilder class.

Issue #4 (User-friendly URLs for RDF representations of resources). To solve this problem, we added support for such URLs by making changes to the ResourceController class.

Issue #6 (Expanding nested resources). The problem was solved as follows. Near URIs of the nested resources, we added “Expand” buttons. When the user clicks on this button, an iframe with the representation of this nested resource is
appended just near its URI (see Fig. 9). Also, we are going to implement more flexible solution, that doesn’t use an iframe.

The descriptions of blank nodes are simply moved from the bottom of the HTML page to the cells with the IDs of the these nodes. For example, Fig. 10 represents the cat synset. For this synset, the value of the wn:definition property is a blank node. This blank node is represented not only by its ID, but also by its description, i.e. a table of its own properties.

**Issue #9 (Support of \LaTeX math notation).** In order to solve this problem, we have developed a special plugin, based on the well-known MathJax library\(^{[18]}\) (see Fig. 11).

![Figure 9. Example of deploying an embedded resource using an iframe.](image)

**CONCLUSION**

We reveal several limitations of LodView that impede its use for navigation over the multilingual Linguistic Linked Open Data cloud. For fixing these limitations, we propose the following improvements: 1) resolution of Cyrillic URIs; 2) decoding Cyrillic URIs in Turtle representations of resources; 3) support of Cyrillic literals; 4) user-friendly URLs for RDF representations of resources; 5) support of hash URIs; 6) expanding nested resources; 7) support of RDF collections; 8) pagination of resource property values; and 9) support of \LaTeX math notation.

At the moment, we have implemented the improvements #1–4 and #9, and partially the improvements #5 and #6. Our fork of LodView is available on GitHub: https://github.com/ManlyMan1/LodView_Cyrillic.

Currently, we are working on the remaining issues. After that, we are going to send a pull request to the main LodView repository.

**Acknowledgment.** The work was initiated for navigation over LLOD cloud at KFU where was funded by RSF according to the project no. 19-71-10056, and then adapted for navigation over specialized \LaTeX-based math collections at JSC RAS.

**REFERENCES**

[1] Christian Bizer, Tom Heath, and Tim Berners-Lee. Linked Data – The Story So Far. *International Journal on Semantic Web and Information Systems*, 5(3):1–22, July 2009. doi:10.4018/jswis.2009081901.

[2] Tom Heath and Christian Bizer. *Linked Data: Evolving the Web into a Global Data Space*. Morgan & Claypool, 2011. URL: http://linked databook.com/editions/1.0/.

[3] Aidan Hogan. *The Web of Data*. Springer, 2020. doi:10.1007/978-3-030-51580-5.

[4] John Philip McCrae et al. The Open Linguistics Working Group: Developing the Linguistic Linked Open Data Cloud. In Nicoletta Calzolari et al., editors, *Proceedings of the 10th International Conference on Language Resources and Evaluation (LREC’16)*, Portoroz, Slovenia, May 23–28, 2016, pages 2435–2441. European Language Resources Association, 2016. URL: https://www.aclweb.org/anthology/L16-1386.

[5] Phillipp Cimiano, Christian Chicarcos, John P. McCrae, and Jorge Gracia. Linguistic Linked Data: Representation, Generation and Applications. Springer, 2020. doi:10.1007/978-3-030-30225-2_3.

[6] Alexander Kirillovich, Olga Nevzorova, Emil Gminadiev, and Natalia Lukachevitch. RuThes Cloud: Towards a Multilevel Linguistic Linked Open Data Resource for Russian. In Pzemyslaw Rózowski and Christoph Lange, editors, *Proceedings of the 8th International Conference on Knowledge Engineering and Semantic Web (KESW 2017)*, Szczecin, Poland, November 8–10, 2017, Communications in Computer and Information Science, vol. 786, pages 38–52. Springer, 2017. doi:10.1007/978-3-319-69548-8_4.

[7] Alfiya Galieva, Alexander Kirillovich, Bulat Khakimov, Natalia Lukachevitch, Olga Nevzorova, and Dzhavdet Suleymanov. Toward Domain-Specific Russian-Tatar Thesaurus Construction. In Radomir Bolgov et al., editors, *Proceedings of the International Conference IMS-2017*, St. Petersburg, Russia, 21–24 June 2017, ACM International Conference Proceeding Series, pages 120–124. ACM Press, New York, 2017. doi:10.1145/3143693.3143718.

[8] Alexander Kirillovich, Marat Shakhkov, Alfiya Galieva, Olga Nevzorova, Dmitry Ilvovsky, and Natalia Lukachevitch. TatWordNet: A Linguistic Linked Open Data-Integrated WordNet Resource for Tatar. In Dagmar Gromann et al., editors, *Proceedings of the 3rd Conference on Language, Data and Knowledge (LDK 2021)*, Zaragoza, Spain, 1–3 Sep 2021, OpenAccess Series in Informatics, vol. 93, pages 16:1–16:12. Schloss Dagstuhl, 2021. doi:10.4230/OASIcs.LDK.2021.15.
