To build densely connected Web of RDF data*

Yasunori Yamamoto[1,*,†], Takatomo Fujisawa[2,‡]

[1]Database Center for Life Science, 178-4-4 Wakashiba, Kashiwa, Chiba 277-0871, JAPAN
[2]National Institute of Genetics, 1111 Yata, Mishima, Shizuoka 411-8540, JAPAN

Abstract

RDF data show their values the most when built in a distributed manner and linked to each other from several aspects with URIs as the keys. However, we have seen several URI mismatches that should be identical from case discrepancies to misuse of symbols such as '#' and '_'. Therefore, RDF curation is needed to make RDF data more linkable and valuable. Here, we propose an infrastructure for RDF data constructors to curate them.

Keywords

RDF, Web of Data, Data curation

1. Introduction

The attempt to express huge and diverse life science data in Resource Description Framework (RDF) has begun since the late 2000s, and the number of newly built RDF data is increasing even now. Currently, 62 SPARQL endpoints are listed at the Umaka-Yummy Data[1] in which you can learn the status of each endpoint such as how stable it is, how fast it returns a result, and so on. RDF demonstrates its maximum potential when each URI denotes one concept and vice versa since a URI is a global identifier. Multiple RDF datasets built in a distributed manner can be easily joined if this is true. However, there are several URI discrepancies among them. In addition to the synonymous URI issue, of which we should take care, these include the following examples.

- `http://www.w3.org/2000/01/rdf-schema#Label`
- `http://www.w3.org/2000/01/rdfschema#label`

We consider that these are due to the nature of a distributed way of building RDF datasets. Multiple people and institutions are involved in building. Therefore, we need not only call community's attention, but also construct an infrastructure to minimize these discrepancies as much as possible with the help of machines. Here, we propose such an infrastructure where RDF data constructors can curate their data effectively and efficiently.

14th International SWAT4HCLS Conference, Feb 13 – 16, 2023, Basel, Switzerland
*Corresponding author.
†These authors contributed equally.
yy@dbcls.rois.ac.jp (Y. Yamamoto); tf@nig.ac.jp (T. Fujisawa)
https://researchmap.jp/yayamamo (Y. Yamamoto); https://researchmap.jp/takatomo (T. Fujisawa)
0000-0002-6943-6887 (Y. Yamamoto); 0000-0001-8978-3344 (T. Fujisawa)

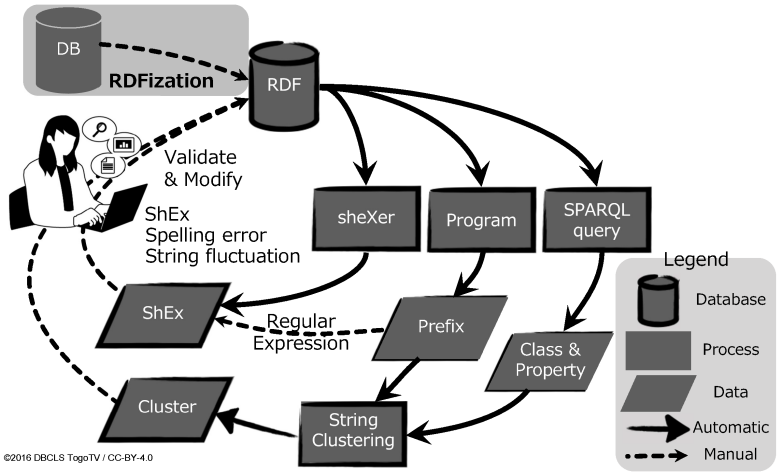

Figure 1: Overall architecture of RDF curation infrastructure

2. RDF data curation infrastructure

Figure 1 describes an overall architecture of RDF curation infrastructure. We assume that RDF data constructors use in-house tools to build an RDF dataset. Therefore, there are some non-RDF data as sources for a target RDF dataset. Since an RDF dataset constructed by these tools is often not what you expect at first, a couple of times you need to repeat the cycle of checking data, modifying code, and generating data. In this cycle, ShEx that conforms to the generated RDF data helps to find errors, and the tool sheXer[2] has this role, that is, to generate ShEx from a given RDF data. In addition, sheXer reports some statistics such as what percentage of instances of a specific class has one specific predicate as a comment in ShEx. We can notice whether there is an outlier or not by looking at them along with the ShEx itself. As there are tools having a function to validate RDF data by a given ShEx such as Apache Jena, regenerated RDF data can be verified to follow the ShEx, which a curator modifies after generated by sheXer. In addition, to find typos in prefixes and URIs of classes or properties, we use a string clustering algorithm such as the fingerprint method.

Acknowledgments

This work was supported under the Life Science Database Integration Project, NBDC of Japan Science and Technology Agency.

References

[1] Y. Yamamoto, A. Yamaguchi, A. Splendiani, Yummydata: providing high-quality open life science data, Database (Oxford) 2018 (2018). doi:https://doi.org/10.1093/database/bay022.

[2] Automatic extraction of shapes using shexer, Knowledge-Based Systems 238 (2022) 107975. doi:https://doi.org/10.1016/j.knosys.2021.107975.