RMLdoc: Documenting Mapping Rules for Knowledge Graph Construction

Jhon Toledo[1], Ana Iglesias-Molina[1], David Chaves-Fraga[2], and Daniel Garijo[1]

[1] Ontology Engineering Group, Universidad Politécnica de Madrid, Spain
{ja.toledo,ana.iglesiasm,daniel.garijo}@upm.es
[2] Grupo de Sistemas Intelixentes, Universidade de Santiago de Compostela, Spain
david.chaves@usc.es

Abstract. In this demo we present RMLdoc, a Python package designed to generate documentation for RML mappings when constructing knowledge graphs from heterogeneous sources. Given an input mapping file written in R2RML, RML, or YARRRML, RMLdoc will generate a detailed Markdown documentation explaining each mapping with corresponding diagrams, in a human readable manner. Thanks to RMLdoc, we aim to shed light in the knowledge graph construction process, making mappings easier to maintain and understand by knowledge engineers.
Code repository: https://github.com/oeg-upm/rmldoc/
Demo: https://w3id.org/rmldoc/example

Keywords: Documentation · Knowledge Graph Construction · RML.

1 Introduction

Knowledge graphs (KGs) are commonly constructed by transforming a set of heterogeneous data sources (e.g., CSV, JSON files) into RDF graphs. These transformations are performed by relating all input sources with the target ontology terms, and can be described using declarative mapping languages such as the W3C recommendation R2RML[3] or its widely adopted extension RML [7]. Institutions such as the European Railway Agency[4] or the European Commission (e.g., in the EU Public Procurement Data Space[5]) describe their transformations using these languages in some of their projects.

Knowledge engineers are usually responsible for developing the mapping rules needed to construct KGs. In many cases, these engineers rely on graphical interfaces (e.g, RMLEditor [5]) and human-friendly serializations like YARRRML [4] or Mapeathor [6] to aid them in the creation of mapping rules. However, the mapping documents resultant from these efforts are in many cases complex and hard to interpret, which reduces their reusability by other engineers. Furthermore, there is a lack of tools to generate a comprehensive and human-readable

[3] https://www.w3.org/TR/r2rml/
[4] https://data-interop.era.europa.eu/
[5] https://europa.eu/!qx9WxQ

documentation of mapping rules. This situation delegates mappings as second-class resources in the KG development process, without documentation (scattered comments in the mapping document at most) or essential metadata (e.g., version, creators, license).

In this paper, we present RMLdoc [8],[6] an open source Python package designed to create a human-readable documentation of the mapping rules used to construct a Knowledge Graph. RMLdoc supports mapping rules described in R2RML, RML, and YARRRML, helping practitioners better understand the relationships between the original data sources and the ontology terms. To the best of our knowledge, this is the first approach that proposes the generation of human-readable mapping documentation. RMLdoc is one step closer towards completing technological support of KG-driven ecosystems.

2 Mapping Documentation with RMLdoc

RMLdoc processes R2RML, RML and YARRRML mappings to generate a human-readable documentation as follows:

Mapping upload and processing. The tool takes as input an existing mapping written in R2RML, RML or YARRRML. In the case of receiving YARRRML, these mappings are first translated into RML using Yatter.[7] The mappings documents as RDF graphs are validated to check for grammar errors, and then loaded internally as a graph. RMLdoc supports both the original proposal of RML [3] and the specification lately developed by the Knowledge Graph Construction W3C Community Group [7].

Querying and information extraction. The mapping graph is then queried to extract the relevant information for its documentation: (i) metadata, (ii) namespaces and (iii) mapping rule sets. First, the *metadata* of the mapping document is queried. This information is optional in the mapping, but recommended for improving its documentation (e.g., description, authors, creation date, license). We retrieve this information taking a mapping document as a `dcat:Dataset` or `schema:Dataset` [2]. Next, the *namespaces* and prefixes declared in the document are extracted, followed by the elements that compose the *mapping rule sets* (in RML, *Triples Map*). From each rule set RMLdoc extracts the data source, subject and predicate-object description, and the joins performed to create triples with references between different rule sets (in RML, *Join Conditions*).

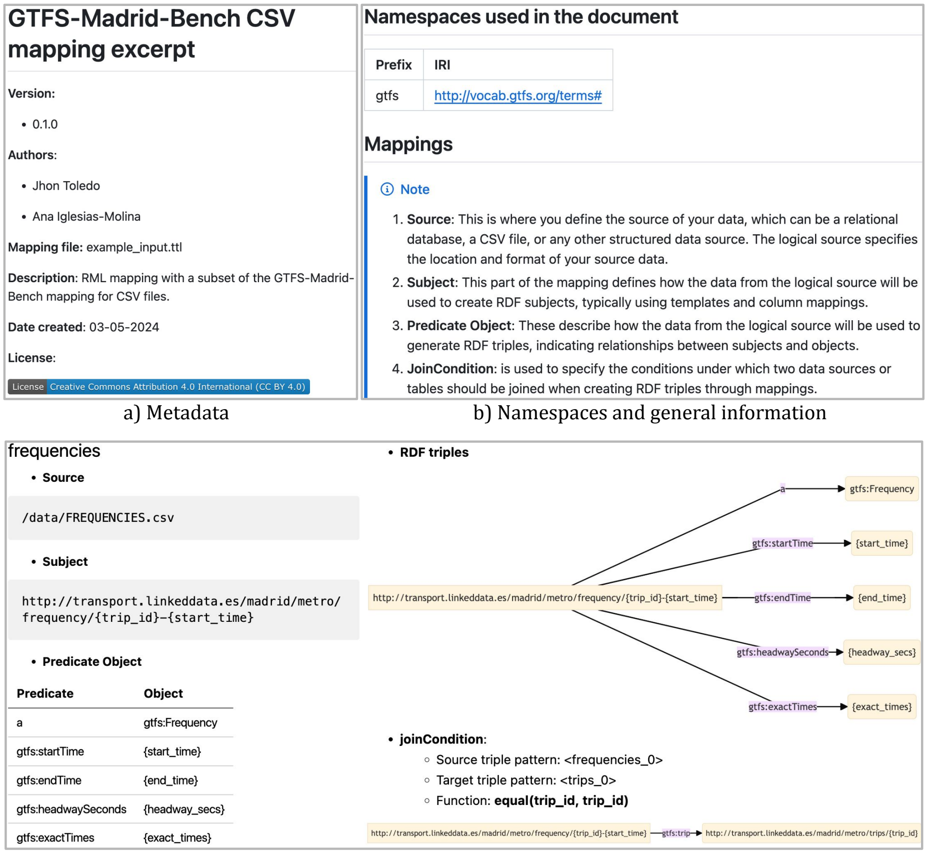
Serialization and writing. The information retrieved in the previous step is structured and written using Jinja templates[8] in a Markdown document to generate the human-readable documentation. Additionally, the triples and joins documented in each rule set are accompanied with a diagram, automatically generated with the Mermaid library.[9]

[6] https://pypi.org/project/rmldoc/

[7] https://pypi.org/project/yatter/

[8] https://jinja.palletsprojects.com/en/2.10.x/templates/

[9] https://mermaid.js.org/



GTFS-Madrid-Bench CSV mapping excerpt

Version:

- 0.1.0

Authors:

- Jhon Toledo
- Ana Iglesias-Molina

Mapping file: example_input.ttl

Description: RML mapping with a subset of the GTFS-Madrid-Bench mapping for CSV files.

Date created: 03-05-2024

License:

License Creative Commons Attribution 4.0 International (CC BY 4.0)

a) Metadata

Namespaces used in the document

Prefix	IRI
gtfs	http://vocab.gtfs.org/terms#

Mappings

ⓘ Note

1. **Source**: This is where you define the source of your data, which can be a relational database, a CSV file, or any other structured data source. The logical source specifies the location and format of your source data.
2. **Subject**: This part of the mapping defines how the data from the logical source will be used to create RDF subjects, typically using templates and column mappings.
3. **Predicate Object**: These describe how the data from the logical source will be used to generate RDF triples, indicating relationships between subjects and objects.
4. **JoinCondition**: is used to specify the conditions under which two data sources or tables should be joined when creating RDF triples through mappings.

b) Namespaces and general information

frequencies

- **Source**

```
/data/FREQUENCIES.csv
```

- **Subject**

```
http://transport.linkeddata.es/madrid/metro/
frequency/{trip_id}-{start_time}
```

- **Predicate Object**

Predicate	Object
a	gtfs:Frequency
gtfs:startTime	{start_time}
gtfs:endTime	{end_time}
gtfs:headwaySeconds	{headway_secs}
gtfs:exactTimes	{exact_times}

- **RDF triples**

- **joinCondition**:
  - Source triple pattern: <frequencies_0>
  - Target triple pattern: <trips_0>
  - Function: **equal(trip_id, trip_id)**

http://transport.linkeddata.es/madrid/metro/frequency/{trip_id}-{start_time}
gtfs:trip
http://transport.linkeddata.es/madrid/metro/trips/{trip_id}

c) Triples map components description with visual depictions

Fig. 1: Demo example from https://w3id.org/rmldoc/example

Figure 1 shows a **demo** example documentation for a mapping subset of the GTFS-Madrid-Bench [1], showing how the mapping information is structured in the Markdown file: the mapping metadata (Fig. 1a) including title, version, authors, file name, description, creation date, and license; the prefixes used and a brief conceptual description of the mapping components (Fig. 1b); and a exemplary rule set (`frequencies`, Fig. 1c). The diagram shows the essential mapping elements in a human-friendly manner, adding a visual aid while avoiding introducing constructs from the languages that are not necessary for the comprehension of the transformation rules.

The source code of RMLdoc is openly available under Apache 2.0 license.[10] Following open science best practices, each release automatically generates a dedicated DOI [8]. Additionally, the tool is available in PyPi as a package.[6]

[10] https://github.com/oeg-upm/rmldoc

3 Conclusions and Future Steps

In this paper we present RMLdoc, a Python library designed to generate human-readable documentation for mappings used in declarative knowledge graph construction. This tool processes mapping documents written in either RML, R2RML or YARRRML and produces a Markdown file with the essential information for understanding the transformation rules, also depicting them in visual diagrams. As future steps, we want to extend the tool further to be fully compliant with all modules of the new RML specification, and allow metadata annotation on the *Triples Map* level. We also plan on supporting HTML export and launching the tool as a GitHub action, with the aim of facilitating an effortless documentation during the KG development process. This is the first approach developed for documenting mapping rules for knowledge graph construction, which we believe that it is a necessary step towards the governance of the artifacts involved in KG-driven ecosystems.

References

1. Chaves-Fraga, D., Priyatna, F., Cimmino, A., Toledo, J., Ruckhaus, E., Corcho, O.: Gtfs-madrid-bench: A benchmark for virtual knowledge graph access in the transport domain. Journal of Web Semantics **65**, 100596 (2020)
2. Dimou, A., De Nies, T., Verborgh, R., Mannens, E., Mechant, P., Van de Walle, R.: Automated metadata generation for linked data generation and publishing workflows. In: Workshop on Linked Data on the Web (LDOW@WWW 2016). CEUR Workshop Proceedings, vol. 1593 (2016)
3. Dimou, A., Sande, M.V., Colpaert, P., Verborgh, R., Mannens, E., Van De Walle, R.: RML: A generic language for integrated RDF mappings of heterogeneous data. In: Workshop on Linked Data on the Web (LDOW@WWW 2014). CEUR Workshop Proceedings, vol. 1184 (2014)
4. Heyvaert, P., De Meester, B., Dimou, A., Verborgh, R.: Declarative Rules for Linked Data Generation at your Fingertips! In: ESWC 2018 Satellite Events. vol. 11155, pp. 213–217. Springer, Cham (2018)
5. Heyvaert, P., Dimou, A., Herregodts, A.L., Verborgh, R., Schuurman, D., Mannens, E., Van de Walle, R.: RMLEditor: A Graph-based Mapping Editor for Linked Data Mappings. In: Extended Semantic Web Conference (ESWC 2016). pp. 709–723. Springer (2016)
6. Iglesias-Molina, A., Pozo-Gilo, L., Doña, D., Ruckhaus, E., Chaves-Fraga, D., Corcho, O.: Mapeathor: Simplifying the specification of declarative rules for knowledge graph construction. In: ISWC 2020 Demos and Industry Tracks. CEUR Workshop Proceedings, vol. 2721 (2020)
7. Iglesias-Molina, A., Van Assche, D., et al.: The RML Ontology: A Community-Driven Modular Redesign After a Decade of Experience in Mapping Heterogeneous Data to RDF. In: International Semantic Web Conference (ISWC 2023). pp. 152–175. Springer (2023)
8. Toledo, J., Chaves, D., Iglesias-Molina, A., Garijo, D.: oeg-upm/rmldoc: rmldoc 0.1.1 (Mar 2024). https://doi.org/10.5281/zenodo.10731583