Results for Knowledge Graph Creation Challenge 2024: SDM-RDFizer

Enrique Iglesias[1,2,*], Maria-Esther Vidal[1,2,3]

[1]L3S Research Center, Hannover, Germany
[2]Leibniz University of Hannover, Hannover, Germany
[3]TIB Leibniz Information Centre for Science and Technology, Hannover, Germany

Abstract
The volume of data generated in recent years has increased drastically, necessitating a unified schema to integrate multiple data sources into a single format. The RDF Mapping Language (RML) was developed to define the structure of knowledge graphs (KGs). Over time, various extensions have been introduced to enhance RML's functionality, creating a need for a new specification that consolidates these extensions. Track 1 of the KGCW 2023 Challenge dataset addresses this need by providing a comprehensive set of test cases to ensure that knowledge graph creation engines comply with the updated RML specification. This paper reports on the performance evaluation of SDM-RDFizer using this dataset, highlighting its capabilities and areas for improvement in achieving full RML compliance.

Keywords
Knowledge Graph Creation, Data Integration System, RDF Mapping Languages

1. Introduction

The substantial surge in data volume has led to the increasing use of knowledge graphs (KGs) to integrate multiple data sources in different formats. Consequently, various mapping languages have emerged to define KGs, including R2RML and its extension, RDF Mapping Language (RML) [1]. Both of these languages adhere to the rules established by the Resource Description Framework (RDF)[1]. Over time, new extensions for RML have been developed, adding functionalities like the execution of functions for value transformation (RML+FnO [2]) and the use of RDF-Star (RML-Star [3]). A new specification for RML has been defined to incorporate all these extensions and remove references to R2RML formally.
The Track 1 dataset of the KGCW 2024 Challenge covers many test cases, including basic cases, functions, RML-Star, remote sources, and specific outputs. These test cases ensure that existing KG creation tools can incorporate the new RML specification and achieve full compliance. This report presents the updates needed to integrate the new specification into SDM-RDFizer and evaluates the results using the Track 1 dataset.

Hersonissos'24: Fifth International Workshop On Knowledge Graph Construction, May 27, 2024, Hersonissos, GR
*Corresponding author.
†These authors contributed equally.
iglesias@l3s.de (E. Iglesias); maria.vidal@tib.eu (M. Vidal)
0000-0002-8734-3123 (E. Iglesias); 0000-0003-1160-8727 (M. Vidal)

[1]https://www.w3.org/TR/2004/REC-rdf-primer-20040210/

This paper is organized into three additional sections. Section 2 provides an overview of SDM-RDFizer, including its techniques, data structures, and physical operators for optimizing KG creation. Section 3 details the results of the challenge, including the dataset definition and the necessary improvements to meet the test cases. Finally, Section 4 presents the conclusions and future steps for SDM-RDFizer.

2. SDM-RDFizer

SDM-RDFizer [4] is a KG creation engine that is RML compliant. SDM-RDFizer is comprised of two modules: **Triples Maps Planning** (TMP) and **Triples Maps Execution** (TME). Each module has different data structures that optimize different aspects of the KG graph creation process. TMP determines the execution order for the triples maps (TM) to keep memory usage to a minimum. TME generates KG following the order established by TMP. Multiple novel operators are defined to transform different types of TMs. **Simple Object Map** (SOM) operator executes *rml:template* and *rml:reference*, **Object Reference Map** (ORM) executes parent triples maps, and **Object Join Map** (OJM) executes joins. All generated triples are compared to the corresponding Predicate Tuple Table (PTT) to determine if it is a duplicate and **Dictionary Table** (DT) compress the resources stored in PTT. **Predicate Join Tuple Table** (PJTT) stores the result of executing join.

3. Test Cases of the Knowledge Graph Creation Challenge

Track 1 of the KGCW 2024 Challenge[2] aims to inspire new methods and techniques for incorporating the new RML specification into existing KG creation engines. This dataset comprises five sets of test cases. **RML-Core:** This set includes basic test cases originally defined in the RML test cases[3] to test the compliance of KG creation engines. These cases have been updated to reflect the new specification and utilize CSV, JSON, XML files, and relational databases (MySQL and Postgres) as data sources. **RML-FNML:** This set contains test cases that use functions to transform data, employing a series of pre-defined functions to execute these transformations. **RML-Star:** This set incorporates RDF-Star[4] test cases, generated from the RML-Star test cases[5] in accordance with the new specification. **RML-IO:** This set includes a wide variety of remote data sources such as endpoints, compressed files, and JSON and XML files. It also defines outputs for specific properties in various formats like Turtle, RDF/JSON, JSON-LD, and multiple compressed formats like Zip and Tar. The name reflects its focus on Input and Output. **RML-CC:** This set comprises collections and containers.
This work presents the results of executing RML-Core, RML-FNML, RML-Star, and RML-IO with SDM-RDFizer. RML-CC will be incorporated at a later date. Table 1 show the total number of test cases in the dataset and which cases were passed and failed by SDM-RDFizer.

[2]https://zenodo.org/records/10973433
[3]https://rml.io/test-cases/
[4]https://www.w3.org/2021/12/rdf-star.html
[5]https://zenodo.org/records/6518802

Set	# of Test Cases	# of Passed Cases	# of Fail Cases
RML-Core	238	238	0
RML-FNML	14	14	0
RML-Star	18	18	0
RML-IO	67	65	2
RML-CC	29	0	29
Total	366	335	31

Table 1
Test Cases of the KGCW 2024 Challenge Track 1 dataset

3.1. Results of RML-Core

RML-Core comprises test cases covering the fundamentals of RML, such as the definition of classes, the use of *rml:template* and *rml:reference*, the execution of parent triples maps and joins, and the definition of data types, languages, and graphs. SDM-RDFizer implements three operators to execute different types of mappings.
To extract data from various data sources, SDM-RDFizer employs several Python libraries: *csv* for CSV files, *json* for JSON files, *xml* for XML files, *mysql-connector* for connecting to MySQL databases, and *psycopg2* for connecting to Postgres databases.
To parse the new specification, the parser query is updated to replace the *rml* prefix with its new namespace, replace all mentions of the R2RML namespace with the RML namespace, and update the definition of *rml:logicalSource* to include the use of the *rml:path* and *rml:root* clauses, replacing *rml:query* with *rml:iterator*. The *rdflib* library is used to execute the parser query.
SDM-RDFizer successfully performs all 238 test cases from this set.

3.2. Results of RML-FNML

RML-FNML consists of test cases that use functions to transform values, including tasks like replacing strings, transforming strings to lower and upper case, concatenating strings, and more. These test cases demonstrate the use of RML+FnO in the new specification. SDM-RDFizer incorporates strategies from FunMap [5] to execute functions. FunMap is a TM translator that converts TMs containing functions and their corresponding data sources into TMs without functions, reflecting the execution of the functions.
To parse TMs with functions, a new parser query is explicitly defined for extracting the functions from the TMs. This allows for proper handling of nested functions, as each function is extracted individually, and nested functions are called from the function's parameters.
SDM-RDFizer successfully performs all 14 test cases from this set.

3.3. Results of RML-Star

RML-Star comprises test cases that use RDF-Star, an extension of RDF that introduces a new term, the quoted triple, which can be used as either the subject or the object of a triple. Therefore, RML-Star presents *rml:quotedTriplesMap*, enabling the definition of quoted triples in a KG. SDM-RDFizer implements a new operator to generate quoted triples, allowing for recursive application since quoted triples can contain other quoted triples.

Another challenge in these test cases was using joins in the *rml:subjectMap*. These cases were handled similarly to joins in the *rml:objectMap*, using the OJM operator for join execution and PJTT for storing the results. The parser query was expanded to recognize *rml:quotedTriplesMap*. SDM-RDFizer successfully performs all 18 test cases from this set.

3.4. Results of RML-IO

RML-IO consists of test cases that cover a wide range of remote data sources, including compressed files, JSON and XML files, and data extracted from SPARQL endpoints. Some cases introduce the concept of outputting certain triples to specific output files, which may need to be compressed or translated into different formats, such as JSON-LD, Turtle, etc.
SDM-RDFizer uses the *requests* library to collect data from remote sources. For SPARQL endpoints, the *SPARQLWrapper* library connects and executes SPARQL queries, with the results converted to a format similar to CSV. When dealing with compressed files, SDM-RDFizer downloads the file locally and decompresses it using the appropriate library for the format, such as the *zip* library for Zip files.
These test cases introduce the concept of defining an alternate output within the TM. These can be specified in the *rml:subjectMap*, *rml:predicateMap*, *rml:objectMap*, *rml:languageMap*, *rml:datatypeMap*, or *rml:graphMap*. Based on its location, triples will be outputted to the alternate output file. If defined in the *rml:subjectMap*, all valid triples generated from this TM will be sent to that particular output. If the alternate output is defined elsewhere, only triples with that specific property will be outputted there. Any remaining triples not destined for an alternate output will be sent to the original output file, which is defined at the start of the SDM-RDFizer execution. When handling these alternative output files, SDM-RDFizer prioritizes generating them over the standard output and can compress them. Finally, SDM-RDFizer converts the output files into various RDF formats, such as RDF/XML, JSON-LD, etc., using the *rdflib* library. SDM-RDFizer successfully executes 65 of the 67 test cases. The two failures occurred because SDM-RDFizer cannot upload the generated triples into a SPARQL endpoint.

4. Conclusions

The KGCW 2024 Challenge Track 1 dataset evaluates the compliance of state-of-the-art engines with the new formulation of RML. It comprises 366 test cases across five categories: RML-Core, RML-IO, RML-FNML, RML-Star, and RML-CC. SDM-RDFizer successfully executes 333 of these test cases, fully covering RML-Core, RML-FNML, RML-Star, and all but two from RML-IO.
To achieve this, SDM-RDFizer introduced a new parsing query, strategies from FunMap, and an operator for generating inner triples for RML-Star. Moving forward, SDM-RDFizer aims to address the remaining RML-CC cases by incorporating new methods. Additionally, a new data structure will be developed for intermediate results in RML-Star to avoid repeated inner triple generation, and an optimized parser will be implemented to manage the increasing complexity. With these improvements, SDM-RDFizer is set to become a fully compliant RML engine.

Acknowledgments

This work has been partially supported by the Federal Ministry for Economic Affairs and Energy of Germany (BMWK) in the project CoyPu (project number 01MK21007[A-L]). Leibniz Association partially funds Maria-Esther Vidal in the "Leibniz Best Minds: Programme for Women Professors", project TrustKG-Transforming Data in Trustable Insights with grant P99/2020.

References

[1] A. Dimou, M. Vander Sande, P. Colpaert, R. Verborgh, E. Mannens, R. Van de Walle, RML: A Generic Language for Integrated RDF Mappings of Heterogeneous Data, in: Workshop on Linked Data on the Web, 2014.

[2] B. De Meester, A. Dimou, R. Verborgh, E. Mannens, An ontology to semantically declare and describe functions, in: European Semantic Web Conference, Springer, 2016, pp. 46–49.

[3] T. Delva, J. Arenas-Guerrero, A. Iglesias-Molina, Ó. Corcho, D. Chaves-Fraga, A. Dimou, Rml-star: A declarative mapping language for rdf-star generation, in: O. Seneviratne, C. Pesquita, J. Sequeda, L. Etcheverry (Eds.), Proceedings of the ISWC 2021 Posters, Demos and Industry Tracks: From Novel Ideas to Industrial Practice co-located with 20th International Semantic Web Conference (ISWC 2021), Virtual Conference, October 24-28, 2021, volume 2980 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2021. URL: https://ceur-ws.org/Vol-2980/paper374.pdf.

[4] E. Iglesias, S. Jozashoori, D. Chaves-Fraga, D. Collarana, M.-E. Vidal, SDM-RDFizer: An RML Interpreter for the Efficient Creation of RDF Knowledge Graphs, in: CIKM, 2020. doi:10.1145/3340531.3412881.

[5] S. Jozashoori, D. Chaves-Fraga, E. Iglesias, M. Vidal, Ó. Corcho, Funmap: Efficient execution of functional mappings for knowledge graph creation, in: J. Z. Pan, V. A. M. Tamma, C. d'Amato, K. Janowicz, B. Fu, A. Polleres, O. Seneviratne, L. Kagal (Eds.), The Semantic Web - ISWC 2020 - 19th International Semantic Web Conference, Athens, Greece, November 2-6, 2020, Proceedings, Part I, volume 12506 of *Lecture Notes in Computer Science*, Springer, 2020, pp. 276–293. URL: https://doi.org/10.1007/978-3-030-62419-4_16. doi:10.1007/978-3-030-62419-4_16.