2021-12

Extending R2RML-F to support dynamic datatype and language tags

Aparna Nayak
Technological University Dublin, d19125691@mytudublin.ie

Bojan Bozic
Technological University Dublin, bojan.bozic@tudublin.ie

Luca Longo
Technological University Dublin, luca.longo@tudublin.ie

Recommended Citation

Nayak, A., Božić, B., & Longo, L. (2021). Extending R2RML-F to support dynamic datatype and language tags. Elsevier. DOI: 10.21427/XVHP-6G08

Funder: Science Foundation Ireland

Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 192 (2021) 709–716

Procedia Computer Science

www.elsevier.com/locate/procedia

25th International Conference on Knowledge-Based and Intelligent Information & Engineering Systems

Extending R2RML-F to support dynamic datatype and language tags

Aparna Nayak [a,*], Bojan Božić[a], Luca Longo[a]

[a]*ML Labs, Technological University Dublin, Dublin, Republic of Ireland*

Abstract

Linked data is often generated from raw data with the help of mapping languages. Complex data transformation is one of the essential parts while uplifting data which either can be implemented as custom solutions or separated from the mapping process. In this paper, we propose an approach of separating complex data transformations from the mapping process that can still be reusable across the systems. In the proposed method, complex data transformations include the entailment of (i) language tag and (ii) datatype present at the data source. The proposed method also includes inferring missing datatype information. We extended R2RML-F to handle data transformations. The results showed that transformation functions could be used to create typed literals dynamically. Our approach is validated on the test cases specified by the RDF mapping language (RML). The proposed method considers data in the form of JSON, thus making the system interoperable and reusable.

Keywords: Knowledge graphs ; Linked data ; Mapping language ; Typed literals

1. Introduction

The meaning of the term ‘Knowledge Graph’ has evolved with the announcement of the Google Knowledge Graph. It has positively impacted the semantic uplifting of different structures of data appearing in heterogeneous formats. Therefore, various approaches have been proposed to generate knowledge graphs from existing (semi-)structured data [18]. One of the approaches involves using direct mappings from (semi-)structured data to the Resource Description Framework (RDF) format. However, the resultant graph of direct mapping is just a direct graph rather than a knowledge graph. Direct mapping is a special case of Relational Database (RDB) to RDF Mapping Language(R2RML) [17]. The W3C standard R2RML is a widely used language for specifying customised mappings needed to generate RDF datasets from relational databases. These mappings are in the form of rules. Detaching the rules renders them interoperable between implementations while the systems that process those rules are independent of the use case. Uplifting data

* Corresponding author. Tel.: +91-9653211634
E-mail address: aparna.nayak@tudublin.ie

1877-0509
10.1016/j.procs.2021.08.073

from any form to RDF requires the selection of appropriate vocabulary. It helps to achieve the objective of the Semantic Web, which is to connect all the entities for reuse across application, enterprise, and community.

RDF [7] stores the data in the form of triples (subject, predicate, and object). Subject and predicate in RDF are identified by Internationalized Resource Identifier (IRI). Objects are represented in the form of literals. Objects with literals can be annotated with optional type information, such as datatype and optional language subtags, to describe the language used to denote an object. A datatype is a classification of data that describes types of RDF literals and are adopted from XML Schema [15]. XML schema supports two classes of datatypes: simple and complex. Simple datatypes can be primitive or derived. Each simple datatype can be uniquely addressed via a Uniform Resource Identifier (URI). Annotating RDF literal with language helps to match and integrate RDF documents [5]. Language tags help to identify the language of the written content. The use of correct language tag helps in translation tools, accessibility, page rendering, and search.

Many dedicated mapping languages were proposed for RDF mapping. However, to the best of our knowledge, a mapping language that supports the annotation of language tag and datatype, which is already present in the dataset as a separate tuple, does not exist. In this paper, we annotate literals with an appropriate datatype and language tags that are already provided in the (semi-)structured dataset. In the case of the missing datatype, it is inferred [5] by considering core datatypes [16]. We aim to answer the following research question:

To what extent can the R2RML-F mapping technique be extended to support annotation of datataype and language tag by incorporating separate complex data transformations from the mapping process?

The answer to the above-mentioned research question is explored in section 3. R2RML [17] is W3C standardized mapping language that also supports creation of typed literals. However, typed literals consider only hard-coded value for the datatype or language tag. The requirement is to override the behaviour of a typed literal creation. The proposed solution supports dynamic annotation of language tag and datatype to the literal.

The aim of this paper is to shed some light on the dynamic creation of typed literals. Particularly, we will explore annotating datatype and language tags given as a separate field in JSON file. In section 2 we will briefly review existing mapping methods to uplift data from various formats to RDF and support functional mapping to allow dynamic annotation. Section 3 we present the design and methodology used to undertake our study. In section 4 we discuss our findings with various test cases. Finally, in section 5 we provide an overview of the conclusions derived from the study and suggest possible future directions of the experiment.

2. Related work

R2RML is a mapping language to express customized mappings from relational databases to RDF datasets. These mappings allow us to represent the relational database that conforms to a schema in RDF format. The main goal of R2RML is to map relational databases to RDF datasets directly. An advantage of R2RML over direct mapping is a mapping author can define highly customized views over relational data. RDB-to-RDF mapping has been an active field of research for which several contributions have been made in the last several years. Significant number of approaches have been defined to uplift heterogeneous sources into RDF such as xR2RML [14], RDF data generation from SPARQL [13], D2RML [1], ShExML [8]. These languages either extends R2RML or makes use of SPARQL to generate RDF datasets. A comparative study to evaluate uplift tools applied to Comma Separated Values (CSV) was presented in a survey article [12].

R2RML supports ‘Transformation function’ [9], that allows to represent a literal value in different syntactic representation. However, these transformation considers underlying database technology that allows required conversion. There exists a couple of functional transformation mapping languages built on R2RML : R2RML-F [2], RML [10]. R2RML-F is an extended version that captures functions in mapping. It also allows the uplift of CSV files into RDF [11]. The functions are executed with the help of ECMAScript; hence the absence of functionality by underlying technology does not make a difference to the core work.

On the other hand, R2RML can be extended altogether to add more functionality such as support of different data sources, serialization formats and many more. One such mapping language is RML. R2RML is a subset of RML that aims to extend applicability [4]. RML is a generic mapping language defined to express customized mapping rules

from heterogeneous data structures and serializations to the RDF data model [3]. However, at this stage, we choose to extend R2RML-F [2] to validate our ideas and consider RML at a later stage.

3. Design and methods

R2RML-F [2] is an extension of W3C-standard mapping language R2RML that supports user-defined function. User-defined functions are useful to define functional transformation in any part of the RDF structure. The proposed method is an extension of R2RML-F. Alongside, it also supports uplifting data from CSV to RDF. R2RML-F requires a function name and a function body to declare a function. The function name and name of the function in the function body must be identical. The function call contains a function name and parameters that have to be processed. Each function must have exactly one function name and exactly one function body. The function body is processed to compute the output, and the value is returned to the called function. Any error with the function output will be sent back to the user.

Figure 1 represents the basic flow diagram of the R2RML-F engine that takes one input file, either a CSV file or an RDB file, one mapping file and the output format from the user to generate RDF data. The mapping file contains rules in turtle format that maps the input file to RDF. This also includes a user-defined function that is responsible for annotating the dynamic value of the datatype/language subtag to the respective value.

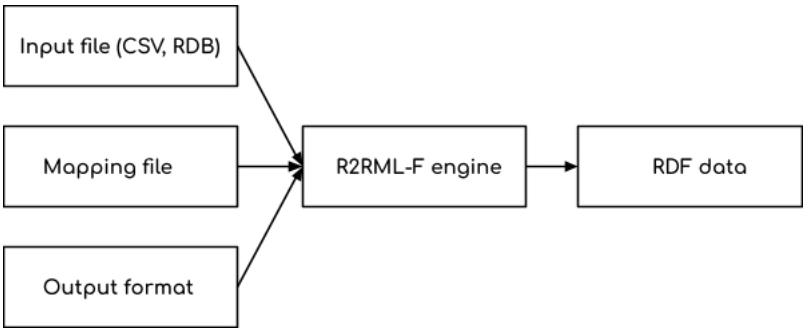

Fig. 1. Flow diagram of extended R2RML-F

To illustrate the concept of mapping and uplifting with an example, consider a file that has single person information in JSON format as shown in figure 2. The preprocessing engine is responsible for converting the data format from JSON to CSV. It also brings the language tags and datatype in the required format. R2RML requires datatype in the form of "xsd:integer" for integer and similar notation hold for other datatypes. Likewise, language tags should be a two-character word that is converted and validated by preprocessing engine. With the help of the mapping file, the extended R2RML-F engine uplifts CSV into RDF.

Complex data transformations remain out of scope for mapping languages. One such concern is the annotation of literal with language tag and datatype. In RDF, typed literal comprises a literal value and a datatype or a language tag. Typed literal allows annotation of literal with datatype and language tag. Figure 3 shows the syntax to generate typed literal in Apache Jena framework [1]. However, this function does not allow the parameterization of the second parameter.

```
model.createTypedLiteral(value, datatype)
```

Listing 1: Creation of typed literal

Two special cases are added to the existing R2RML-F functionality. The first case is the annotation of a datatype and the second case is the annotation of a language tag. Function in R2RML-F is defined as shown in figure 3. In the example, 'attachdt' is a function name. When the function is called, its body will get executed, and a value is returned.

[1] https://jena.apache.org/

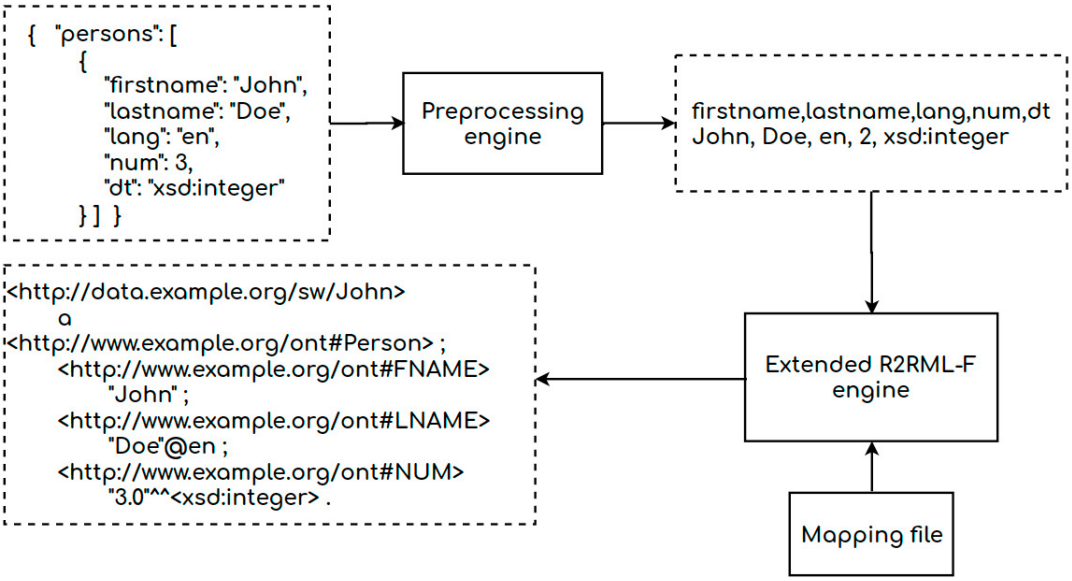

Fig. 2. Illustrative example of extended R2RML-F

The datatype and language tag that are present in the data source is rendered as a column. Therefore, the column that holds literal values and the column that holds datatype value must be specified in parameter bindings. Figure 4 depicts a function call from R2RML object map. All the numbers that belong to column 'NUM' are annotated with corresponding values present in the 'DT' column. When the function name is neither 'attachdt' nor 'attachlg', the system executes the body of the function and returns the appropriate value. Additionally, data is preprocessed to get the data in the required format before converting it into RDF format. The steps followed in preprocessing are discussed in the following section 4.

```
<#Attachdt>
rrf:functionName "attachdt" ;
rrf:functionBody """
//Omitted
""" ;
.
```

Fig. 3. Function declaration in R2RML-F

```
rrf:functionCall [
rrf:function <#Attachdt> ;
rrf:parameterBindings (
  [rr:column "NUM"]
  [rr:column "DT"]
) ;
] .
```

Fig. 4. Function in a Predicate Object Map

4. Experiment and Result

We demonstrate and evaluate our approach with test cases specified by RML [2]. RML test cases are designed to verify the mapping language, and there are no test cases to support the annotation of datatype and language tag. Test cases are modified to support language tag and datatype. Furthermore, all the use cases provided by knowledge graph construction workshop [3] for annotating datatype and language tags are considered. Altogether the proposed method is verified against 39 test cases.

[2] https://rml.io/test-cases/
[3] https://github.com/kg-construct/mapping-challenges/tree/main/challenges

JSON data files are preprocessed to get all datatypes and language tags in the required format for mapping. The preprocessing steps include the steps as shown in algorithm 1. The algorithm considers JSON data as input, preprocesses the data to the required format and outputs the comma-separated values. The preprocessing steps make use of a stack to convert all JSON key-value pair into a list. This list later saved as a comma-separated value which becomes an input to the R2RML-F engine. Before pushing any value into the stack, it is also conformed to a well-formed datatype. Missing datatype leads to the inference of datatype based on the recently pushed item into the stack.

Algorithm 1: Data preprocessing

```
Result: Comma separated values of data
Read JSON data. Initialize a stack.
while all key-value pairs do
    Push the key value pair into the stack.
    if key=='dt' then
        Pop the value. ;
        if value!='xsd:nnn' then
            Update the value to 'xsd:nnn' format.
        end
        if value==' ' then
            Predict the type based on recent pushed item.
        end
    end
    Push the value into the stack.
end
Separate key-value pairs into two different list.
Identify repeated patterns in key list to calculate total number of rows and columns.
Generate dataframe based in total number of rows and columns.
if languageTag header then
    Replace all complete language words into tag.
end
Create a comma separated values using dataframe.
```

The value of the datatype that we are interested in is in the form of "xsd:nnn". Here, nnn can be any RDF-compatible core datatype [16]. Missing datatypes are predicted either based on regular expression or based on JSON type. For example, datatype value for "xsd:integer" can be in the form of int, integer, or "http://www.w3.org/2001/XMLSchema#integer" irrespective of the sensitivity of the case. All the forms are finally converted into "xsd:integer".

JSON objects are created by identifying a repeated pattern of the keys, which represents the header of CSV data. All the values are converted into a data frame based on the number of elements identified in the header. Most language tags consist of a two or three letter language subtag. Language subtags are updated based on the following two scenarios.

- **Language tag is given:** Value that language subtag considers are in the form of 'en' for english, 'fr' for french, 'ga' for irish and so on. These values can be directly substituted to create typed literals.
- **Language is given:** When the language tag is given in the form of full word, it is replaced with corresponding tag. All the languages and its corresponding tags are scraped from W3 schools [4].

[4] https://www.w3schools.com/tags/ref_language_codes.asp

Table 1. Evaluation scenarios of extended R2RML-F

Scenario	Language tag	Datatype
Single object	Yes	Missing datatype
Single object	Yes	Yes
Single object	Yes	No
Single object	No	Missing datatype
Single object	No	Yes
Single object	No	No
Multiple objects	Yes	Missing datatype
Multiple objects	Yes	Yes
Multiple objects	Yes	No
Multiple objects	No	Missing datatype
Multiple objects	No	Yes
Multiple objects	No	No

Once we have all the datatype and language tag in the required format, the entire data frame is written as comma-separated values. Mapping for each test case is written in turtle. Furthermore, data is converted using extended R2RML-F. Table 1 shows the different scenarios that have been considered for the evaluation process. The first column in the table represents the number of JSON objects in each test case. Language tag and datatype column represents the inclusion of the same in the test case. JSON data is stored in the form of objects. The code is made publicly available at github repository [5].

One special scenario where languages are described as separate JSON object is also implemented. In such cases, it is required to name the JSON head as “languages” to differentiate among the normal objects. Some of the limitations of the research are

- In case of multiple object scenario, total items in each object should be of same length.
- Keys in the JSON should be a single word.
- Keys cannot be null values.
- Headers of the datatype column and language column should be declared as ‘dt’ and ‘lang’.

The preprocessed data is used to generate RDF triples using extended R2RML-F. R2RML-F engine returns typed literals when the function is named as “attachdt” or “attachlg”. Each input file requires a mapping file associated with it. Based on the customized mapping, data is converted into RDF triple format. Generated RDF triples are validated using the W3C RDF validation service [6] [6]. This helps to understand the generated RDF data is parsable by any system.

Figure 5 shows sample input considered, figure 6 shows preprocessed output, and figure 7 shows data in RDF TURTLE format. The example includes multiple scenarios such as missing datatype, language tag as a complete language name, multiple objects. Missing datatypes are inferred, language tags are converted into the required format to generate RDF that depicts annotated data.

Our experiment shows that the R2RML-F mapping technique can be extended to support the annotation of datatype and language tags without complex data transformation. We have used user-defined functions of R2RML-F by changing the function definitions to support dynamic annotation of datatype and language tags because it enhances the readability of RDF.

[5] https://github.com/aparnanayakn/r2rmlpreprocess
[6] https://www.w3.org/RDF/Validator/

```
{
  "persons": [
    {
      "firstname": "John", "lastname": "Doe",
      "lang": "english", "num": 3, "dt": " "
    },
    {
      "firstname": "Jane", "lastname": "Smith",
      "lang": "fr", "num": "3.14", "dt" : " "
    }
  ]
}
```

Fig. 5. Sample input in JSON format

```
firstname,lastname,lang,num,dt
John,Doe,en,3.0,xsd:integer
Jane,Smith,fr,3.14,xsd:decimal
```

Fig. 6. Prepossessed output in CSV format

```
<http://data.example.org/sw/John>
      a      <http://www.example.org/ont#Person> ;
      <http://www.example.org/ont#FNAME>
            "John" ;
      <http://www.example.org/ont#LNAME>
            "Doe"@en ;
      <http://www.example.org/ont#NUM>
            "3.0"^^<xsd:integer> .

<http://data.example.org/sw/Jane>
      a      <http://www.example.org/ont#Person> ;
      <http://www.example.org/ont#FNAME>
            "Jane" ;
      <http://www.example.org/ont#LNAME>
            "Smith"@fr ;
      <http://www.example.org/ont#NUM>
            "3.14"^^<xsd:decimal> .
```

Fig. 7. Prepossessed output in RDF format

5. Conclusions and Future work

This research presents a simple approach to map JSON data sources into RDF and to create typed literals using extended R2RML-F. The proposed method follows the principle of separating complex data transformations from the mapping process. Our method efficiently solves the research questions discussed in section 1. The proposed model is evaluated using RML test cases by adding the datatype and language tag in the input file. The model's extensibility is self-evident as the whole solution is separated from the mapping process. R2RML-F treats function calls as a term map. These mapping function helps to create typed literals in RDF.

Future work includes creating typed literals using heterogeneous data structures and RML. It also includes carrying out experiments to validate our findings and develop additional scenarios. The use of RML helps to maintain all items in each object of the same length which is the current limitation of the system. Further, RML also supports mapping data from multiple sources and different format into linked data. The system can also be extended to assess data quality which will ensure their usability.

Acknowledgements

This publication has emanated from research conducted with the financial support of Science Foundation Ireland under Grant number 18/CRT/6183. For the purpose of Open Access, the author has applied a CC BY public copyright licence to any Author Accepted Manuscript version arising from this submission.

References

[1] Chortaras, A., Stamou, G., 2018. Mapping diverse data to rdf in practice, in: International Semantic Web Conference, Springer. pp. 441–457.

[2] Debruyne, C., O'Sullivan, D., 2016. R2rml-f: Towards sharing and executing domain logic in r2rml mappings, in: Linked Data on the Web, CEUR Workshop Proceedings.

[3] Dimou, A., Sande, M., Colpaert, P., Verborgh, R., Mannens, E., Van De Walle, R., 2014. Rml: A generic language for integrated rdf mappings of heterogeneous data, CEUR Workshop Proceedings.

[4] Dimou, A., Vander Sande, M., Colpaert, P., Mannens, E., Van de Walle, R., 2013. Extending r2rml to a source-independent mapping language for rdf, in: International Semantic Web Conference (Posters & Demos), pp. 237–240.

[5] Dongo, I., Cardinale, Y., Chbeir, R., 2018. Rdf-f: Rdf datatype inferring framework: Towards better rdf document matching. Data Science and Engineering 3, 115–135. doi:10.1007/s41019-018-0064-6.

[6] Eric Prud'hommeaux, Ryan Lee, T.G., 2007. W3c rdf validation service. W3C recommendation .

[7] Frank Manola, E.M., 2004. Rdf primer. W3C recommendation .

[8] García-González, H., Boneva, I., Staworko, S., Labra-Gayo, J.E., Lovelle, J.M.C., 2020. Shexml: improving the usability of heterogeneous data mapping languages for first-time users. PeerJ Computer Science 6, e318.

[9] Hert, M., Reif, G., Gall, H., 2011. A comparison of rdb-to-rdf mapping languages. ACM International Conference Proceeding Series , 25–32doi:10.1145/2063518.2063522.

[10] Jozashoori, S., Chaves-Fraga, D., Iglesias, E., Vidal, M.E., Corcho, O., 2020. Funmap: Efficient execution of functional mappings for knowledge graph creation, in: International Semantic Web Conference, Springer. pp. 276–293.

[11] Junior, A., Brennan, R., Debruyne, C., O'Sullivan, D., 2016. Funul: A method to incorporate functions into uplift mapping languages, Association for Computing Machinery. pp. 267–275. doi:10.1145/3011141.3011152.

[12] Junior, A.C., Debruyne, C., Brennan, R., O'Sullivan, D., 2017. An evaluation of uplift mapping languages. International Journal of Web Information Systems .

[13] Lefrançois, M., Zimmermann, A., Bakerally, N., 2017. A sparql extension for generating rdf from heterogeneous formats, in: European Semantic Web Conference, Springer. pp. 35–50.

[14] Michel, F., Djimenou, L., Zucker, C.F., Montagnat, J., 2015. Translation of relational and non-relational databases into rdf with xr2rml, in: 11th International Confenrence on Web Information Systems and Technologies (WEBIST'15), pp. 443–454.

[15] Paul V. Biron, A.M., 2004. Xml schema part 2: Datatypes second edition. W3C recommendation .

[16] Richard Cyganiak, David Wood, M.L., 2014. Rdf 1.1 concepts and abstract syntax. W3C Recommendation .

[17] Souripriya Das, Seema Sundara, R.C., 2012. R2rml: Rdb to rdf mapping language. W3C Recommendation .

[18] Spanos, D.E., Stavrou, P., Mitrou, N., 2012. Bringing relational databases into the semantic web: A survey. Semantic Web 3, 169–209.