Incorporating Functions in Mappings to Facilitate the Uplift of CSV Files into RDF

Ademar Crotti Junior(✉), Christophe Debruyne, and Declan O'Sullivan

ADAPT Centre for Digital Content Technology Research, Knowledge and Data Engineering Group, School of Computer Science and Statistics, Trinity College Dublin, Dublin 2, Ireland
{crottija,debruync,declan.osullivan}@scss.tcd.ie

Abstract. Many solutions have been developed to convert non-RDF data to RDF. A common task during this conversion is applying data manipulation functions to obtain the desired output. Depending on the data format of the source to be transformed, one can rely on the underlying technology, such as RDBMS for relational databases or XQuery for XML, to manipulate data - to a certain extent - while generating RDF. For CSV files, however, there is no such underlying technology. Instead, one has to resort to more elaborate Extract, Transform and Load (ETL) processes, which can render the generation of RDF more complex (in terms of number of steps), and therefore also less traceable and transparent. One solution to this problem is the declaration and inclusion of functions in mappings of non-RDF data to RDF. In this paper, we propose a method to incorporate functions into mapping languages and demonstrate its viability in Digital Humanities use case.

Keywords: Linked Data · Mapping · Data manipulation

1 Introduction

The CSV file format is a convenient and popular way to exchange structured data, but the semantics of the information captured in such files are not explicit. RDF, on the other hand, provides one means to add meaning to data that software can process. The process of converting data in any non-RDF format (e.g., CSV and relational databases) into RDF is called *uplift*.

There are scenarios where it is necessary to manipulate data during the uplift process. Depending on the source data format, this can be very straightforward. With uplift languages for relational databases such as R2RML[1], for example, one can rely on the underlying RDBMS to support some data manipulation tasks (e.g., string concatenation). The same is true for XSPARQL [5], where one can use XQuery to manipulate the data contained in XML. Sometimes, however, relying on the underlying technology is not sufficient [4]. For CSV datasets there is no such equivalent, standardized underlying technology. When data in CSV files has to be manipulated to generate RDF, one needs to resort to data pre- or post-processing. This increases complexity in terms of number

[1] https://www.w3.org/TR/r2rml/.

of steps necessary to generate RDF, and renders the whole data processing "pipeline" less transparent. A solution to tackle this problem is to capture these manipulations as functions in the mappings.

We propose a method to incorporate functions into mapping languages that draws inspiration from, and generalizes ideas presented in [4]. To demonstrate our method, we extend RML's vocabulary and engine to include notions for function calls and parameter bindings. The main contributions of this paper can be summarized as follows: (i) a method to incorporate functions in a mapping language; (ii) an implementation of the method extending RML; and (iii) a demonstration of functions incorporated into mappings applied to a real world dataset.

2 Related Work

One approach to support uplift is to use an annotation language to relate non-RDF data to RDF (i.e., "mappings"), for which an engine is built. Examples of this approach include R2RML and R2RML-F [4] for relational databases; SML [3] for relational databases and CSV; and RML [1] and KR2RML [7] for an even wider array of non-RDF data formats. These mapping languages usually have access to functionality provided by the underlying technology of the non-RDF data source. For CSV files, these do not exist and one has to apply data pre- or post-processing techniques, which raises problems as explained in Sect. 1.

To the best of our knowledge, KR2RML is the only tool to support data manipulation functions inside a mapping language that does not rely on the underlying technology. Though they provide an editor in which you can load data and input mappings to create functions in Python to manipulate that data, once those functions are stored several problems can be observed. First, a lot of the structured information containing the function is captured as a string. This thus requires both parsing the file and that string. Secondly, the mapping becomes rather complex, which makes it more difficult for users to create similar mappings with other tools. Their editor, however, does facilitate the mapping creation process for their mapping language. In [4] an extension to R2RML called R2RML-F is proposed. R2RML-F adds supports for capturing domain knowledge inside the mapping language for relational databases. Unlike KR2RML, functions in R2RML-F are captured as resources *referred to* by mappings with the RDF data model, allowing functions to be reused in different mappings.

3 Incorporating Functions into Mapping Languages

In this section, we describe how we adopt ideas presented in [4] to develop a more generic, usable and amenable approach to incorporate functions into mapping languages. These functions can be used to capture both domain knowledge (e.g., transforming units) and other – more syntactic – data manipulation tasks (e.g., transforming values to create valid URIs). Function names are unique and each function must have one function name and one function body. A function body defines a function with a return statement; parameters are optional.

Our proof-of-concept extends RML's vocabulary and engine[2] by introducing construct for describing functions, function calls and parameter bindings. Listing 1 defines a function. This function has one string as a parameter and returns a URL concatenated with its camel case version. Although this function executes a simple string transformation, functions in this method are generic and capable of complex data transformations. Furthermore, functions work with any data format and can be reused in the mapping. Listing 2 demonstrates how the function is called. In R2RML – and by consequence RML – a Term Map generates an RDF term (see [1]). In our implementation, we introduced a new Term Map called a *Function Valued* Term Map that generates RDF terms based on the application of a function. The parameters are also Term Maps that are evaluated before the results are passed as arguments.

```
<#Camelize>
 rrf:functionName "camelize" ;
 rrf:functionBody """ function camelize (str) { var camelCaseString =
str.toLowerCase().replace( /[-_]+/g, ' ').replace( /[^\\w\\s]/g, ' ').replace( /
(.)/g, function($1) { return $1.toUpperCase(); }).replace( / /g, '' );
return "http://dacura.cs.tcd.ie/data/seshat/" + camelCaseString; } """ ; .
```

Listing 1: Declaring a function

```
<#Variable>
 rml:logicalSource [ rml:source "data.csv"; rml:referenceFormulation ql:CSV ];
 rr:subjectMap [ rr:termType rr:BlankNode; ];
 rr:predicateObjectMap [
   rr:predicateMap [ rrf:functionCall [ rrf:function <#Camelize> ;
       rrf:parameterBindings ( [ rml:reference "Variable" ] );]; ];
  rr:objectMap [ rr:parentTriplesMap <#Value> ] ].
```

Listing 2: Calling a function in a PredicateMap

4 Demonstration

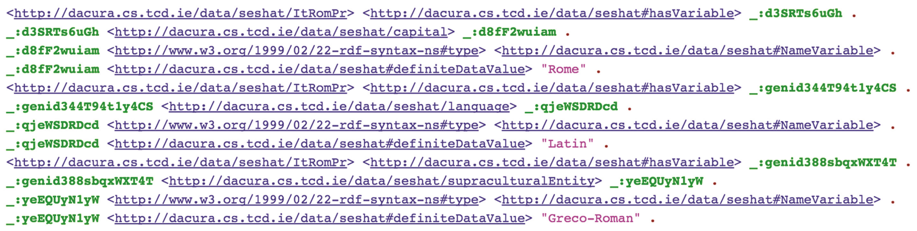
The dataset used to demonstrate our approach comes from the Seshat: Global History Databank [6]. This project is developing a knowledge base to describe human history that is created by hand via a wiki where contributors are expected to adhere to certain conventions to structure the facts. The Seshat dataset is currently made available for analysis as CSV files by scraping the wiki pages. A current development within the project is to gather the data into an OWL knowledge base, but predicates from the dataset differ from the predicates defined in the OWL ontology. Thus, there is a need to develop an approach to transform CSV values into the URIs of the ontology's predicates.

For example, in the dataset one predicate is labeled as "Capital", but in the ontology the predicate is *seshat:capital*. Other examples include "Language" and "Supracultural entity". Table 1 shows a fragment of the data where the values for the column "Variable" are transformed into predicates using the mapping from Listings 1 and 2. Example RDF output is shown in Fig. 1.

[2] Available at https://github.com/CNGL-repo/RMLProcessor.

Table 1. Excerpt of the CSV file shown as a table.

NGA	Polity	Section	Variable	Value from	Fact type	Value note
Latium	ItRomPr	General variables	Capital	Rome	Simple	Simple
Latium	ItRomPr	General variables	Language	Latin	Simple	Simple
Latium	ItRomPr	General variables	Supracultural entity	Greco-Roman	Simple	Simple



```
<http://dacura.cs.tcd.ie/data/seshat/ItRomPr> <http://dacura.cs.tcd.ie/data/seshat#hasVariable> _:d3SRTs6uGh .
_:d3SRTs6uGh <http://dacura.cs.tcd.ie/data/seshat/capital> _:d8fF2wuiam .
_:d8fF2wuiam <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://dacura.cs.tcd.ie/data/seshat#NameVariable> .
_:d8fF2wuiam <http://dacura.cs.tcd.ie/data/seshat#definiteDataValue> "Rome" .
<http://dacura.cs.tcd.ie/data/seshat/ItRomPr> <http://dacura.cs.tcd.ie/data/seshat#hasVariable> _:genid344T94t1y4CS .
_:genid344T94t1y4CS <http://dacura.cs.tcd.ie/data/seshat/language> _:qjeWSDRDcd .
_:qjeWSDRDcd <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://dacura.cs.tcd.ie/data/seshat#NameVariable> .
_:qjeWSDRDcd <http://dacura.cs.tcd.ie/data/seshat#definiteDataValue> "Latin" .
<http://dacura.cs.tcd.ie/data/seshat/ItRomPr> <http://dacura.cs.tcd.ie/data/seshat#hasVariable> _:genid388sbqxWXT4T .
_:genid388sbqxWXT4T <http://dacura.cs.tcd.ie/data/seshat/supraculturalEntity> _:yeEQUyN1yW .
_:yeEQUyN1yW <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://dacura.cs.tcd.ie/data/seshat#NameVariable> .
_:yeEQUyN1yW <http://dacura.cs.tcd.ie/data/seshat#definiteDataValue> "Greco-Roman" .
```

Fig. 1. RDF output.

5 Conclusions and Future Work

Most tools to convert data to RDF rely on underlying technology for data manipulation, but there is no manipulation language for CSV datasets. Moreover, when data manipulation is needed for CSV datasets one depends on pre- or post-processing techniques, which adds complexity to the uplift process. One solution is to incorporate functions in mappings, but the state-of-the-art does not offer a feasible way to do so. We tackled this problem by presenting a more amenable method to incorporate functions into mapping languages. We demonstrated our approach by extending RML's vocabulary and engine and applied it on a real world dataset.

Future work includes more use cases and experiments to compare performance and expressiveness of our approach and other mapping languages. Since functions can be considered software agents, one can also generate provenance information referring to these functions. Inspiration can be drawn from [2], who proposed a method for creating provenance information while generating RDF.

Acknowledgements. This study is supported by: (i) CNPQ, National Counsel of Technological and Scientific Development – Brazil; (ii) the Science Foundation Ireland ADAPT Centre for Digital Content Technology (Grant 13/RC/2106); (iii) John Templeton Foundation grant to the Evolution Institute [https://evolution-institute.org/project/seshat/]; (iv) the European Union Horizon 2020 ALIGNED [www.aligned-project.eu] (Grant 644055).

References

1. Dimou, A., Vander Sande, M., 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 (2014)
2. Dimou, A., De Nies, T., Verborgh, R., Mannens, E., and Van de Walle, R.: Automated metadata generation for Linked Data generation and publishing workflows. In: Workshop on Linked Data on the Web (2016)
3. Stadler, C., Unbehauen, J., Westphal, P., Sherif, M.A., Lehmann, J.: Simplified RDB2RDF mapping. In: Workshop on Linked Data on the Web (2015)
4. Debruyne, C., O'Sullivan, D.: R2RML-F: towards sharing and executing domain logic in R2RML mappings. In: Workshop on Linked Data on the Web (2016)
5. Bischof, S., Decker, S., Krennwallner, T., Lopes, N., Polleres, A.: Mapping between RDF and XML with XSPARQL. J. Data Semant. **1**(3), 147–185 (2012)
6. Turchin, P., Brennan, R., Currie, T., Feeney, K., Francois, P., Hoyer, D., Manning, J., Marciniak, A., Mullins, D., Palmisano, A., et al.: Seshat: the global history data-bank. Cliodynamics J. Quant. Hist. Cult. Evol. **6**, 77–107 (2015)
7. Slepicka, J., Yin, C., Szekely, P., Knoblock, C.: KR2RML: an alternative interpretation of R2RML for heterogeneous sources. In: Proceedings of the 6th International Workshop on Consuming Linked Data (2015)