Sustainable Linked Data Generation: The Case of DBpedia

Wouter Maroy[1], Anastasia Dimou[1](✉), Dimitris Kontokostas[2], Ben De Meester[1], Ruben Verborgh[1], Jens Lehmann[3,4], Erik Mannens[1], and Sebastian Hellmann[2]

[1] imec – IDLab, Department of Electronics and Information Systems, Ghent University, Ghent, Belgium
{wouter.maroy,anastasia.dimou,ben.demeester,ruben.verborgh, erik.mannens}@ugent.be

[2] Leipzig University – AKSW/KILT, Leipzig, Germany
{Kontokostas,Hellmann}@informatik.uni-leipzig.de

[3] University of Bonn, Smart Data Analytics Group, Bonn, Germany
jens.lehmann@cs.uni-bonn.de, jens.lehmann@iais.fraunhofer.de

[4] Fraunhofer IAIS, Sankt Augustin, Germany

Abstract. DBpedia EF, the generation framework behind one of the Linked Open Data cloud's central interlinking hubs, has limitations with regard to quality, coverage and sustainability of the generated dataset. DBpedia can be further improved both on *schema* and *data* level. Errors and inconsistencies can be addressed by amending (i) the DBpedia EF; (ii) the DBpedia mapping rules; or (iii) Wikipedia itself from which it extracts information. However, even though the DBpedia EF and mapping rules are continuously evolving and several changes were applied to both of them, there are no significant improvements on the DBpedia dataset since its limitations were identified. To address these shortcomings, we propose adapting a different semantic-driven approach that decouples, in a declarative manner, the *extraction*, *transformation* and mapping rules *execution*. In this paper, we provide details regarding the new DBpedia EF, its architecture, technical implementation and extraction results. This way, we achieve an enhanced data generation process, which can be broadly adopted, and that improves its quality, coverage and sustainability.

1 Introduction

The *DBpedia Extraction Framework* (DBpedia EF) [12] extracts raw data from Wikipedia and makes it available as Linked Data, forming the well-known and broadly used DBpedia dataset. The majority of the DBpedia dataset is derived through *Wikipedia infobox templates*, after being annotated by the *DBpedia*

The described research activities were funded by Ghent University, imec, Flanders Innovation and Entrepreneurship (AIO), the Fund for Scientific Research Flanders (FWO Flanders), and the EU's H2020 program for ALIGNED (GA 644055).

C. d'Amato et al. (Eds.): ISWC 2017, Part II, LNCS 10588, pp. 297–313, 2017.
DOI: 10.1007/978-3-319-68204-4_28

ontology[1] [12]. The DBpedia dataset is further enriched with additional information derived from the articles such as free text, abstracts, images, links to other Web pages, and tables. *Mapping Templates*, i.e., *rules* generating most of the DBpedia dataset from Wikipedia are executed by the DBpedia EF, defined by a world-wide crowd-sourcing effort, and maintained via the *DBpedia mappings wiki*[2].

Even though DBpedia is one of the central interlinking hubs in the Linked Open Data (LOD) cloud [16], its generation framework has limitations that reflect on the generated dataset [14,18]. We distinguish two types of issues with DBpedia:

schema-level rules that define *how* to apply vocabularies to raw data [6, 10,14], e.g., the `dbo:militaryBranch` property is used for entities of `dbo:MilitaryUnit` type, but it should only be used with entities of `dbo:Person` type [13].

data-level *extracted, or processed and transformed data values.* The former includes incorrect, incomplete or irrelevant extracted values and datatypes, or not (well-)recognized templates [18]; the latter issues with parsing values [17], interpreting/converting units [17], or transforming cardinal direction [15].

Errors or inconsistencies in DBpedia can be addressed by amending the DBpedia EF, mapping rules, or Wikipedia itself [18]. However, even though several changes were applied to both the DBpedia EF and mapping rules [12], quality issues still persist. For instance, 32% of its mapping rules are involved in at least one inconsistency [13], and more than half of all cardinal direction relations are still invalid or incorrect [15]. These issues challenge DBpedia dataset's usage, hence the demands for DBpedia EF to improve in terms of *expressivity*, i.e., level of details that a user can specify in mapping rules, *semantic flexibility*, i.e., possibility to interchange among different schemas, and *modularity*, i.e., declarative rules and modular components for extraction, transformation and mapping. Given that quality issues persist over a long period of time [13,15], the custom DBpedia EF appears to not meet the above and more foundational changes are required.

In this work, we show that the coupling among *extraction*, i.e., retrieving data from Wikipedia, *transformation*, i.e., processing the extracted data values, and *mapping rules execution*, i.e., applying semantic annotations to the retrieved and transformed data values, contributes to DBpedia EF's inadequacy to cope with the increasing demands for high quality Linked Data. While just decoupling these processes and keeping them hard-coded is just (re-)engineering, we look into a radical solution that turns the DBpedia EF more semantic rather than just a black-box. The goal is to adjust the current DBpedia EF and provide a general-purpose and more sustainable framework that enables more added value. For instance, in terms of semantic flexibility, interchanging among different schema

[1] http://dbpedia.org/ontology/.
[2] http://mappings.dbpedia.org/index.php/Main_Page.

annotations should be supported, instead of e.g., being coupled to a certain ontology as it is now, and allowing only certain semantic representation.

In this paper, we show how we incorporate in the existing DBpedia EF a general-purpose semantic-driven Linked Data generation approach, based on RML [7]. It replaces the current solution, decouples extraction, transformations and conditions from DBpedia EF, and enables generating high quality Linked Data for DBpedia. The activities to achieve this solution started as a GSoC2016 project[3,4] and continue till nowdays. Our contribution is three-fold: we (i) outline the limitations of the current DBpedia EF; (ii) identify the requirements for a more sustainable approach; and (iii) incorporate a generic approach that fulfills those requirements to the current DBpedia EF and compare the two approaches.

The paper is organized as follows: after outlining the state of the art (Sect. 2), we detail the current DBpedia EF limitations and requirements for a sustainable framework (Sect. 3). In Sect. 4, we introduce our approach, and provide a corresponding implementation. In Sect. 5, we provide the results of our approach's evaluation. Finally, we summarize our conclusions in Sect. 6.

2 Background and Related Work

In this section, we discuss related work in Linked Data generation (Sect. 2.1) and we outline the current DBpedia EF functionality (Sect. 2.2).

2.1 State of the Art

In the past, *case-specific solutions* were established for Linked Data generation, which couple schema and data transformations. XSLT- or XPath-based approaches were established for generating Linked Data from data originally in XML format, e.g., AstroGrid-D[5]. There are also *query-oriented* languages that combine SPARQL with other languages or custom alignments to the underlying data structure. For instance, XSPARQL [1] maps data in XML format and Tarql[6] data in CSV. Nevertheless, those approaches cannot be extended to cover other data sources.

Different approaches emerged to define data transformations declaratively, such as VOLT [15] for SPARQL, Hydra [11] for Web Services, or FnO [4]. Hydra or VOLT depend on the underlying system (Web Services and SPARQL, respectively), thus their use is inherently limited to it. Using Hydra descriptions for executing transformations only works online, whereas VOLT only works for data already existing in a SPARQL endpoint. Describing the transformations using FnO does not include this dependency, thus allows for reuse in other use cases and technologies.

[3] www.mail-archive.com/dbpedia-discussion@lists.sourceforge.net/msg07837.html.
[4] https://summerofcode.withgoogle.com/projects/#6213126861094912.
[5] http://www.gac-grid.de/project-products/Software/XML2RDF.html.
[6] https://tarql.github.io/.

2.2 DBpedia Extraction Framework

The DBpedia Extraction Framework (DBpedia EF) generates the DBpedia dataset [12]. It extracts data from Wikipedia, such as infobox templates or abstracts. The DBpedia EF consumes a *source*, i.e., an abstraction of a set of Wiki pages in wikitext syntax, parses it with a Wiki parser, and transforms it in an Abstract Syntax Tree (AST). To process the page's content, several AST-processing components, called *extractors* (e.g., Mapping Extractor, Geo Extractor), traverse a Wiki page's AST and generate RDF triples. Eventually the DBpedia EF forwards them to a sink that outputs them to different datasets based on their properties.

Extractors. The core DBpedia EF components are the *extractors*, whose common main functionality is to traverse the AST-representation of a Wiki page and generate RDF triples based on the syntax tree and other heuristics. Many extractors contribute in forming the DBpedia dataset [12, Sect. 2.2]. A prominent one is the *Mapping-Based Infobox Extractor*, which uses manually written mapping rules that relate infoboxes in Wikipedia to terms in DBpedia ontology, and generates RDF triples. It is the most important extractor, as the most valuable content of DBpedia dataset is derived from infoboxes [12], and covers the greatest part of the DBpedia ontology. Infobox templates are defined in wiki syntax and summarize information related to a page's article in a structured, consistent format which is presented on the article's page as a table with attribute-value pairs.

Mapping-Based Infobox Extractor. It uses community-provided, but manually written mapping rules, which are available at the DBpedia Mappings Wiki.[7] The mappings wiki enables users to collaboratively create and edit mapping rules, specified in the custom DBpedia Mapping Language. The DBpedia Mapping Language relies on MediaWiki templates to define the DBpedia ontology classes and properties and align them with the corresponding template elements [12]. A mapping rule assigns a type from the DBpedia ontology to entities that are described by the corresponding infobox and the infobox's attributes are mapped to DBpedia ontology properties. This extractor traverses the AST and finds infoboxes for which user-defined mapping rules were created. The infoboxes' attribute-value pairs are extracted and RDF triples are generated.

DBpedia Mapping Language. The custom DBpedia mapping language is in wikitext syntax and is used to define how RDF triples are generated from infobox. Each mapping template contains mapping rules for a certain infobox template, such as geocoordinates and date intervals. The mapping rules might specify another mapping template for an infobox property to help the DBpedia

[7] http://mappings.dbpedia.org.

EF produce high quality Linked Data [12]. The following exemplary mapping document contains three different mapping templates: (i) a class mapping (Listing 1, line 2) that maps articles that use that template to the DBpedia Automobile class, (ii) a property mapping template (line 4), which maps the name attribute in the infobox to `foaf:name` and, (iii) a date interval mapping template (line 7), which splits the production attribute (which should be a start and end date) and semantically annotates each part with different ontology terms.

```
 1  {{TemplateMapping
 2   | mapToClass = Automobile
 3   | mappings =
 4       {{PropertyMapping
 5           | templateProperty = name
 6           | ontologyProperty = foaf:name }}
 7  {{DateIntervalMapping
 8       | templateProperty = production
 9       | startDateOntologyProperty = productionStartDate
10          | endDateOntologyProperty = productionEndDate }} }}
```

Listing 1. Extract of a DBpedia mapping templates in wikitext syntax

3 Limitations and Requirements

To address DBpedia dataset quality issues, the current DBpedia EF limitations need to be addressed by adopting a more sustainable approach. In this section, we discuss the current limitations (Sect. 3.1) and requirements for a sustainable framework (Sect. 3.2) that enables higher quality Linked Data generation.

3.1 Limitations

The current DBpedia EF has the following limitations:

mapping rules and implementation coupling Although a set of DBpedia *Mapping Templates* (DMT) is available, the DBpedia community is limited to this set only and cannot easily introduce new ones. The reason is that the DMTs translation happens directly inside the DBpedia EF. Thus, different mapping rules that are defined in each DMT are coupled to their implementation. Some rules, such as transformation of values are, in some case, more flexible to change. For example, changing the `language` value from 'de' to 'el', or converting a string to an IRI by appending a namespace, can be done directly from the mapping rules. However, there are still many useful mapping rules that are not supported without adjusting the DBpedia EF. For instance, combining different values of an infobox is not supported.

Any community member who would like to adjust the mapping rules should be aware of how to develop the DBpedia EF and Linked Data principles. Extending, adjusting or adding a new template, requires (i) extending the current custom DBpedia mapping language, by specifying and documenting the new constructs, and (ii) providing the corresponding implementation to extract these constructs to generate the corresponding RDF triples.

Extending the language is not straightforward and is performed in an ad-hoc manner, while the corresponding implementation is developed as custom solutions, hampering the DBpedia EF maintenance.

transformations and implementation coupling The DBpedia community can neither adjust nor add new transformations because the mapping rules only refer to schema transformations. Transformations over extracted data values are hard-coded and executed at different places within the DBpedia EF, from *extraction* to *mapping* and RDF *generation*. For instance, the actual value for the birth_date property of Person infobox is {{*Birth date and age*|*yyyy*|*mm*|*dd*}}. The DBpedia EF, parses and extracts this to a valid XSD date value, i.e., `"yyy-mm-dd"`. If another date format is desired, it is required to be implemented within the DBpedia EF. Similarly, for the mapping rules that use the DBpedia ontology, DBpedia EF retrieves the DBpedia ontology predicates ranges and uses the defined range to transform the values accordingly. For example, for object properties, the DBpedia EF tries to extract only links and for datatype properties, custom parsers for numbers, floats, dates etc. are applied. Nonetheless, there are cases where the values can be of different types and the users cannot override this behavior without extending the DBpedia EF. Another limitation is the hard-coded unit measurements calculations which, when combined with not consistently formatted input, can lead to wrong values [17]. Overall, the DBpedia dataset is restricted to certain transformations which cannot be easily amended, unless the DBpedia EF is amended, which on its own turn, is not so trivial. Even reusing existing transformation functions requires adjusting the DBpedia EF.

hard-coded mapping rules The DBpedia community cannot adjust all RDF triples which form the DBpedia dataset, because not all of them are generated based on mapping rules. Certain RDF terms and triples are generated without mapping rules being defined, but the DBpedia EF generates them based on hard-coded mapping rules. Adjusting such mapping rules requires adjusting the DBpedia EF. For instance, each RDF triples's subject is dependent on the context of the extraction's execution. Entities in localized datasets (a DBpedia dataset within a certain language) are identified with a subject that is a language specific IRI, e.g., the http://{lang}.dbpedia.org/resource/{resource} namespace is used in the different DBpedia language editions. Generating RDF terms that represent entities with other identifiers or adding new entities cannot be expressed with a mapping rule. For instance, adjusting the aforementioned IRI template to http://{lang}.dbpedia.org/example/{resource} or generating another entity with a different IRI, such as http://example.com/{resource}, requires adjusting the implementation. Overall, configuring current mapping rules has limited influence on most RDF terms and triples generation.

restricted to the DBpedia ontology The DBpedia community cannot use other schema(s) to annotate the Wikipedia pages, than the DBpedia ontology. The current mapping extractor functions only with DBpedia ontology, e.g., the predicate depends on the ontology term used for a certain attribute of an infobox. This occurs because the DBpedia EF interprets the context and selects the corresponding parser based on where the mapping template is used and which ontology term is selected. For instance, the `dbo:date` triggers the `Data parser`. If an ontology term is not added to the DBpedia ontology, it cannot be used, e.g., only the `dbo:location` may be used to indicate an entity's location. Other vocabularies, such as *geo*[8] vocabulary, cannot be used unless imported into the DBpedia ontology. Incorporating any other vocabulary requires adjusting the DBpedia EF, because the *extractor* will not recognize its properties, namely it will not generate RDF triples if `geo:location` is provided. Only certain vocabularies, such as `dcterms`[9] or `foaf`[10], are supported. Similarly, the assigned data type is also dependent on the mapping template and ontology term, e.g., the area in square kms generates an `xsd:double` but also a DBpedia datatype (`dbo:areaTotal`) that depends on the used predicate.

domain validation If the DBpedia community uses ontology terms which cause violations, there is no support for schema validation. Domain validation of the mapping templates defined in the custom DBpedia mapping language can not be supported [6]. Nevertheless, the DBpedia dataset quality would significantly improve if schema violations is applied to the mapping templates [6,13,18]. Currently custom DBpedia mapping templates are only validated for syntax violations (using the Mapping Syntax Validator[11] [12]).

3.2 Requirements

Adjusting the DBpedia EF with a general-purpose Linked Data generation tool allows to adopt a more sustainable solution and enables generating higher quality Linked Data. Such a sustainable approach has the following requirements:

1. **declarative mapping rules** The DBpedia community needs to be able to directly edit, adjust, and define mapping rules for all RDF triples which are generated, while the underlying implementation should be able to interpret them in each case. Hence, a declarative language is required which is able to express all mapping rules. A corresponding underlying implementation should generate all RDF triples relying on declaratively defined mapping rules, either they refer to *schema* or *data* transformations [5]. If the DBpedia mapping rules are formalized in a generic and complete approach, the DBpedia EF mapping process and maintenance will be improved, and, thus, the DBpedia dataset quality will improve too, as it is already indicated by e.g., [6,13].
2. **modular and decoupled implementation** The DBpedia community should be able to add new mapping rules for schema annotations, and alternate or add new transformation rules, as well as data transformation libraries, without requiring to adjust the underlying implementation. The *extraction*,

8 http://www.w3.org/2003/01/geo/wgs84_pos#.
9 http://purl.org/dc/terms/.
10 http://xmlns.com/foaf/0.1/.
11 http://mappings.dbpedia.org/server/mappings/en/validate/.

transformation, *mapping* and RDF *generation* should be decoupled from each other, but aligned as modular components which can be extended or replaced, if e.g., other or new transformations are desired [5].
3. **machine-processable mapping rules** The DBpedia community should be able to build applications which can automatically process the mapping rules. To achieve this, mapping rules should be processable by both humans and machines [7]. For instance, machine processable mapping rules can be assessed not only for syntax but also for schema validation. More, the results of the validation might be automatically processed, as it occurs e.g., with [6,13], or automated mapping rules generation might occur as indicated by [8,9].
4. **vocabulary-independent** The DBpedia community should be able to generate each time any data model is desired, uniquely identify entities as it is desired, as well as annotate data values derived from Wikipedia pages relying on any vocabulary. Mapping rules should be defined and executed, and RDF terms and triples should be generated, independently of the vocabulary used to annotate the extracted data values.

4 Sustainable DBpedia EF with RML

To overcome the current DBpedia EF limitations, we developed a solution that fulfills the aforementioned requirements. R2RML [2] is a W3C standardized language for defining mapping rules to generate Linked Data from data residing in relational databases. The RDF Mapping Language (RML) [7] extends R2RML [2] to enable specifying how Linked Data is generated from sources in different (semi-)structured formats, such as CSV, XML, and JSON. Mapping rules in RML are expressed as RDF triples *(Requirement 3)* and any vocabulary can be used to annotate the data *(Requirement 4)*. Thanks to its extensibility [7], RML can also cover wikitext syntax to generate Linked Data from Wikipedia *(Requirement 1)*. Moreover, RML is aligned with FnO [4], an ontology to define data transformations declaratively *(Requirement 1)*. The RMLMapper executes mapping rules expressed in RML, while the FnO Processor interprets data transformation expressed in FnO and discovers corresponding libraries that execute them *(Requirement 2)*.

In this section, we discuss how we replaced the existing *Infobox based Mapping Extractor*, which is custom for DBpedia EF, with the RMLMapper[12] and challenges we faced. The transition to the new solution was fulfilled in three steps:

1. **mapping rules translation** The DBpedia mapping rules were translated in RML (schema) and FnO (data transformation) statements (Sect. 4.1);
2. **transformations decoupling** The DBpedia Parsing Functions were decoupled from the DBpedia EF and aggregated in a distinct module (Sect. 4.2);
3. **mapping rules execution** The RMLMapper was integrated as extractor (the RMLExtractor) in the DBpedia EF (Sect. 4.3) to execute the mapping rules.

[12] https://github.com/RMLio/RML-Mapper.

```
1  <#infobox_country_mapping_en> rr:subjectMap <subject_mapping_1>;
2     rml:predicateObjectMap <pom_mapping_1>.
3
4  <subject_map_1> rr:template "http://en.dbpedia.org/resource/{wikititle}".
5  <pom_mapping_1> rr:predicateMap [ rr:constant dbo:name. ];
6    rr:objectMap [ rml:reference "common_name"; rr:datatype xsd:string ].
7
8  dbf:extract-entity a fno:Function ;
9    fno:name    "generates a DBpedia IRI" ;
10   dcterms:description "returns an entity" ;
11   fno:expects  ( [ fno:predicate dbf:property ] ) ;
12   fno:output   ( [ fno:predicate dbf:entity ] ) .
13
14 :exe a fno:Execution ;
15   fno:executes dbf:extractEntity ;
16   dbf:property  "Bill Gates";
17   dbf:entity dbr:Bill_Gates .
```

Listing 2. RML mapping rules and FnO data transformations

4.1 Mapping Rules Translation

To take advantage of the RML-based solution, we needed to translate the custom DBpedia mapping rules into RML statements. This was one of the most labor-intensive tasks of the integration process and it was performed in two phases, firstly the translation and, then, the decoupling. In more details:

1. The custom mapping templates in wikitext syntax were translated in RML statements. The RML mapping rules after the first phase can be found at http://mappings.dbpedia.org/server/mappings/en/pages/rdf/.
2. The mapping rules which were embedded in the DBpedia EF were expressed as RML statements and the data transformations as FnO statements. Every mapping template was assigned its own function, e.g., the *DateInterval Template* was assigned the *DateInterval Function* with same parameters. Moreover, we added basic functions like *ExtractDate*, which extracts and processes dates from a value. The RML mapping rules after this phase can be found at http://mappings.dbpedia.org/rml_mappings-201705.zip.

Manually translating all mapping rules in RML would have been difficult, hence, a temporary extension in the DBpedia EF was built to automate the translation[13]. This extension builds on top of the RMLModel [14], and can be run both within the DBpedia EF and standalone. To generate the new RML mapping rules, the original custom DBpedia mapping rules are loaded by the DBpedia EF. Once loaded, the mapping rules are represented in the DBpedia EF data structures and based on these, RML mapping rules are automatically generated. Each template is translated into RML statements and are dumped to a file.

The RML building blocks are *Triples Maps* (Listing 2: line 1) which define how RDF triples are generated. A *Triples Map* consists of three main parts: the

[13] https://github.com/dbpedia/extraction-framework/tree/rml/server/src/main/scala/org/dbpedia/extraction/server/resources/rml.
[14] https://github.com/RMLio/RML-Model.

Logical Source, the *Subject Map* and zero or more *Predicate-Object Maps*. The *Subject Map* (line 4) defines how unique identifiers (URIs) are generated for the mapped resources and is used as the subject of all RDF triples generated from this *Triples Map*. A *Predicate-Object Map* (line 2) consists of *Predicate Maps*, which define the rule that generates the triple's predicate (line 2) and *Object Maps* (line 6) or *Referencing Object Maps*, which define how the triple's object is generated. The *Subject Map*, the *Predicate Map* and the *Object Map* are *Term Maps*, namely rules that generate an RDF term (an IRI, a blank node or a literal).

The Function Ontology (FnO) [3,4] allows to declare and describe functions uniformly, unambiguously, and independently of their implementation technology. A *function* (`fno:Function`, Listing 2: line 8) is an activity which has input parameters (line 11), output (line 12), and implements certain algorithm(s). A *parameter* is the description of a function's input value. An *output* is the description of its output value. An *execution* (`fno:Execution`, line 14) assigns values to the function's parameters for a certain execution. For instance, `dbf:extractEntity` (line 8) is a function extracted from DBpedia EF and generates a DBpedia IRI (line 17) for a given Wikipedia title (line 16) which is passed as parameter. An *Execution* (line 14) can be instantiated to bind a value to the parameter. The result is then bound to that *Execution* via the `dbf:entity` property (line 17).

Addressed Challenges. While the original DBpedia mapping templates mainly build on top of a core template, different templates emerged to cover specific cases, such as templates for geocoordinates and date intervals. Therefore, we had to make sure that each extension and each edge case is covered and RML mapping rules are automatically generated. Nevertheless, a few cases were deliberately excluded because they were not used or they were introduced only for very rare or specific cases; others were omitted because they did not produce sustainable IRIs (see Sect. 5.1 for details). For the latter, the community still needs to agree on a more sustainable modeling. Moreover, we had to interpret each parameter's underlying functionality and to describe it declaratively. Last, the mapping rules and data transformations which were embedded in the DBpedia EF should also be declaratively described and this required to manually define additional RML mapping rules. Refining and extending the mapping rules required several iterations before we reach to a version that generates the same RDF triples as the current DBpedia EF. Moreover, even though it is convenient that certain RDF statements are generated without the community being involved, the complete lack of control over what is generated or how the data values are transformed, it is often the cause for deteriorating DBpedia dataset quality [17,18]. Then again, declaratively defining everything causes an overhead when editing the mapping rules. Nevertheless, it is not expected that the DBpedia community will directly edit the mapping rules, as a corresponding interface is foreseen (GSoC2017[15]).

[15] https://summerofcode.withgoogle.com/projects/#6205485112885248.

4.2 Transformations Decoupling

The custom DBpedia mapping rules allow the DBpedia community to partially define how Linked Data is generated from Wikipedia (schema transformations), without allowing though to customize how the extracted data values can be transformed (data transformations) to form the desired Linked Data. The data transformations are hard-coded in DBpedia EF and restricted to what is implemented. To this end, after formally describing all DBpedia mapping rules and transformations, we decoupled their implementation from the DBpedia EF.

All template functionalities were extracted and gathered into an independent module which can be found at https://github.com/FnOio/dbpedia-parsing-functions-scala/. The FnO Processor is integrated as an independent module in the RMLMapper. It uses the DBpedia data transformation declarations in FnO to retrieve and execute the corresponding implementation each time.

A function might disclose the functionality of a certain mapping template, e.g., the *Property Mapping Template*[16] becames *Simple Property Function*[17] Moreover, other functions were isolated from the DBpedia EF which performed data transformations without a mapping template being involved. There were three major cases: (i) the *extract-entity* function[18] which takes a value of an infobox and creates a DBpedia IRI from it. For instance, it takes '*Melinda gates*' and returns <http://dbpedia.org/Melinda_Gates>; (ii) the *dates* function which takes a date and generates a valid XSD date; and (iii) the *string* function which removes the wikitext syntax and returns the actual data value. All of these functions my be reused beyond the DBpedia EF scope.

Besides data transformations, there were mapping rules which were embedded in the DBpedia EF and they generated automatically RDF triples, without having corresponding mapping rules, such as the datatype annotations, i.e., http://www.w3.org/2001/XMLSchema#double, as well as the custom DBpedia datatypes, e.g., https://dbpedia.org/ontology/PopulatedPlace/areaTotal.

Addressed Challenges. It was challenging to identify each single transformation that the current DBpedia EF has for transformations, as they were spread all over the implementation. The mapping rules only provide a very abstract and high level idea (e.g., dates or coordinates). Moreover, extracting those transformations and gathering them as an independent module for reuse beyond the scope of DBpedia required thorough analysis of the DBpedia EF implementation. The heuristics of these functions are optimized for a large number of

[16] https://github.com/dbpedia/extraction-framework/blob/0496309a0e142b27d940e9d8baa25446b1da4ccb/core/src/main/scala/org/dbpedia/extraction/mappings/SimplePropertyMapping.scala.

[17] https://github.com/FnOio/dbpedia-parsing-functions-scala/blob/development/src/main/scala/functions/implementations/SimplePropertyFunction.scala.

[18] https://github.com/FnOio/dbpedia-parsing-functions-scala/blob/development/src/main/scala/functions/implementations/core/ExtractEntityFunction.scala.

cases and their reuse would contribute on improving the quality of datasets beyond DBpedia.

4.3 Mapping Rules Execution

The generation procedure of DBpedia EF is *data-driven*. Namely, the Linked Data generation is driven by the data sources and an extract of data is considered each time to perform the applicable mapping rules; in contrast to *mapping-driven* where the mapping rules request for the applicable data.

To make sure that the performance remains good, the mapping rules are pre-loaded. This offers faster lookups among the mapping rules, compared to reloading them every time from the hard-disk. The RML mapping rules are executed by triggering the RMLExtractor. In the beginning of each generation process, the RML-MapDocHandler[19] loads all RML mapping rules which are considered for generating DBpedia in memory. Once all RML mapping templates are loaded, the DBpedia dataset generation is initiated. The data is extracted from all Wikipedia pages and the Extraction Manager forwards the incoming AST to all configured extractors, including the RMLExtractor. The RMLExtractor runs over the data to identify the infobox templates which are included in a certain Wiki page. When an infobox template is found, the corresponding RML mapping rules are identified and once all relevant ones are gathered, the RMLMapper is triggered with the AST and mapping rules as input. For instance, when the RMLExtractor finds an infobox, e.g., `Infobox_person`, the RMLExtractor will load the corresponding RML mapping rules, e.g., `Mapping_en:Infobox_person.rml.ttl`. The infobox associated with each set of RML mapping rules are provided to the RMLMapper which processes these inputs and returns RDF triples. Eventually, the RMLExtractor forwards all generated RDF triples to an output sink.

Addressed Challenges. Developing an RMLExtractor that performs large extractions from a Wikipedia dataset within an acceptable time frame had some complexities. The RMLMapper was not designed for being used as an external module, adaptations had to be made to make the integration with DBpedia EF possible. Additionally, the RMLMapper does not process datasets in parallel, since the DBpedia EF does exploit parallel processing. These obstacles have been addressed, but there is still room for more improvement. Besides performance, there were issues with processing steps (e.g., storing triples in specific datasets) during mapping rules executions. These processing steps were initially executed by the DBpedia EF but were adapted because, with the new approach, the RMLMapper handles the mapping execution, instead of the DBpedia EF.

4.4 New DBpedia Architecture

The DBpedia EF allows adding different extractors to fulfill different Linked Data generation processes. Taking advantage of its modularity, a new extractor was

[19] https://github.com/RMLio/RML-MapDocHandler.

built, called the RMLExtractor. The DBpedia EF with the RMLExtractor can be found at https://github.com/dbpedia/extraction-framework/tree/rml.

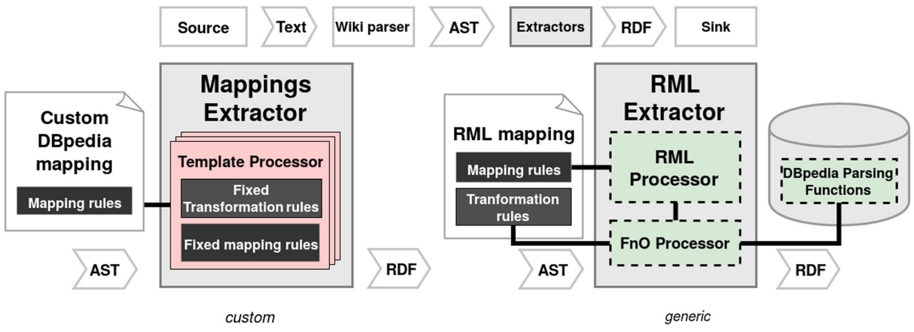

Fig. 1. To the left the Mappings Extractor, to the right the new RML Extractor:

The RMLExtractor, as every other extractor, is available to be used when a user triggers DBpedia's generation. It is independent of other extractors and runs over each AST to generate RDF triples for the infobox values that are contained in the AST based on the loaded RML mapping templates. The RMLExtractor wraps up the RMLMapper which is used as a library (Fig. 1). Each infobox together with the associated mapping template are given as input to RMLMapper, to generate the corresponding RDF triples. The RMLMapper depends on the Function Processor (FnO Processor)[20] which, depends on the DBpedia Parsing Functions[21]. The FnO Processor parses the FnO statements [3] which are included in the DBpedia mapping rules. Afterwards, it fetches the functions and executes them, using values derived from each extraction as parameters. The DBpedia Parsing Functions were extracted from the DBpedia data parsers and were put in an independent module which can be reused beyond the DBpedia EF scope.

5 Evaluation

The new approach was compared with the existing DBpedia EF with respect to *coverage* (Sect. 5.1), *performance* (Sect. 5.2), and *flexibility* (Sect. 5.3).

5.1 Coverage

We generated RDF triples for 16,244,162 pages of the English Wikipedia (version 20170501[22]), both with the current Mapping Extractor and new RMLExtractor.

[20] https://github.com/FnOio/function-processor-java.
[21] https://github.com/FnOio/dbpedia-parsing-functions-fno.
[22] https://dumps.wikimedia.org/enwiki/20170501/enwiki-20170501-pages-articles.xml.bz2.

The former extraction yielded 62,474,123 RDF triples in total, the latter 52,887,156. In terms of entities, 4,747,597 are extracted with the current, whereas 4,683,709 entities are extracted with RMLExtractor, offering 98% coverage.

The subset of RDF triples which overlap was verified to be the same but the RDF datasets differ in size. Even though the mapping rules of all languages were translated in RML statements, certain mapping rules were *deliberately* omitted, due to the following reasons: (i) *unsustainable URIs*; the DBpedia EF itself generates intermediate entities whose URI is generated relying on an iterator, e.g., http://dbpedia.org/resource/Po_(river)__mouthPosition__1. Assigning URIs this way does not generate sustainable identifiers. Therefore, those mapping rules were temporarily omitted, until the DBpedia community decides – now that this can be resolved outside of the DBpedia EF– on how they should be modeled in a more sustainable fashion. (ii) *custom DBpedia datatypes*; the DBpedia EF generates additional RDF triples with custom DBpedia datatypes for units of properties. We omitted them from the RML mapping rules because there is still discussion on whether these should be included in the dataset or not. (iii) *RDF triples added by the DBpedia EF;* These are RDF triples generated from a value of an infobox, if the article's text contains a URI with this value. Currently the RMLMapper does not support referencing the articles text, thus generating those RDF triples is temporarily omitted, until the RMLMapper is configured to work with data from Wikipedia beyond infoboxes.

5.2 Performance

The prototype RMLExtractor generates the DBpedia dataset on reasonable time. Currently, performance is put aside for more sustainable mapping rules and data transformations, as achieving sustainability is much more important than speed which may be optimized in the future. The RMLExtractor is still on average 0.46 ms slower per page than the original framework. Namely the RMLExtractor requires 1.31 ms/page, compared to the original DBpedia EF which requires 0.85 ms/page. On a larger scale the RMLExtractor generates Linked Data from 16,244,162 pages in 5 h and 56 min, compared to the current DBpedia EF which requires 3 h and 50 min (35% slower than the DBpedia EF, Fig. 2).

5.3 Flexibility

The mapping rules, being RDF triples, can be (automatically) updated and other semantic annotations can be applied or other datasets can be generated from Wikipedia. For instance, relying on the DBpedia mapping rules and the alignment of DBpedia with schema.org[23], we translated the RML mapping rules for DBpedia, but any other vocabulary may be used too. This way, we can generate a DBpedia dataset with *schema.org* annotations, instead of only using the DBpedia ontology. This was accomplished by generating a new RML mapping file with *schema.org*

[23] http://schema.org.

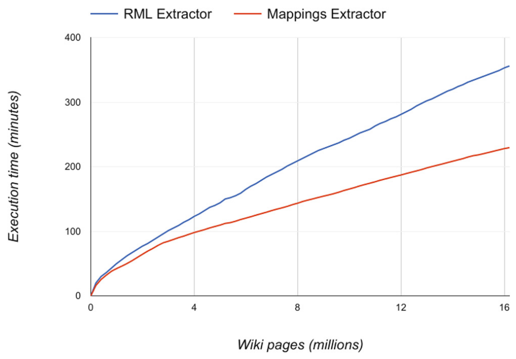

Fig. 2. Performance comparison: Mappings Extractor vs RML Extractor

annotations, based on the mapping rules with the DBpedia ontology annotations. The extracted data values are transformed to a clean format by relying on the same FnO functions which are specified in the RML mapping rules. In our exemplary case, we based on *Infobox_person* mapping template that generates RDF triples for entities of person type. An extraction was done over 16,244,162 pages, 191,288 Infobox_persons were found. 1,026,143 RDF triples were generated. Indicatively, 179,037 RDF triples were generated with `schema:name` property, 54,664 with `schema:jobTitle`, 23,751 with `schema.org:nationality`, 144,907 with `schema.org:birthPlace`, and 139,488 with `schema.org:birthDate`. The dataset is available at http://mappings.dbpedia.org/person_schema.dataset.ttl.bz2.

6 Conclusions and Future Work

In this paper, we presented a generic and semantics-driven approach to replace the Mapping Extractor in DBpedia EF which contributes in the adoption of Semantic Web technologies and Linked Data principles. Thereby, we address several major challenges that currently exist in the DBpedia generation process. We show that our solution can cover same Linked Data output with a more sustainable and reusable process, without prohibitively increasing the execution time.

The resulting new DBpedia EF is easier to maintain, as changes are limited to declarative mapping and transformation rules. We moved from a tightly coupled architecture to a sustainable Linked Data generation approach, as the code of the extractor does not require modifications anymore to improve the Linked Data generation. This better prepares DBpedia for changes, and thereby also facilitates quality improvements to the output. Indeed, many errors and inconsistencies in DBpedia are either due to the old DBpedia EF [18]—which our solution succeeds—and due to the DBpedia ontology [13]—which can more easily be adjusted [13] or replaced thanks to our solution. More, we can apply validation on the mappings themselves, catching potential inconsistencies even before they are generated [6].

Because of the above reasons, the DBpedia community will apply our solution as the default setup for generating DBpedia in the future. The source code is already available on the official DBpedia GitHub repository in a separate branch.

Importantly, and in contrast to the old DBpedia EF, the solution we propose is not specific to DBpedia and can therefore be applied to other use cases as well. Any part of the Linked Data generation can be reused to generate other datasets, such as the mapping and transformation rules; or the transformation functions that were written as small, reusable units. As this functionality becomes shared with other generation workflows, any improvement to them will directly lead to improvements in others. Sustainability thereby spreads beyond just DBpedia.

This work gives rise to several opportunities for future work. A user interface is planned and will enable people to collaboratively create, update, and manage DBpedia generation, while assessing the impact of new ontologies becomes straightforward. By improving the DBpedia generation sustainability, we not only improve DBpedia today, but enable continuous advancements in the future.

References

1. 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)
2. Das, S., Sundara, S., Cyganiak, R.: R2RML: RDB to RDF mapping language. Working group recommendation, W3C, September 2012. http://www.w3.org/TR/r2rml/
3. De Meester, B., Dimou, A.: The Function Ontology. Unofficial Draft (2016). https://w3id.org/function/spec
4. De Meester, B., Dimou, A., Verborgh, R., Mannens, E.: An ontology to semantically declare and describe functions. In: Sack, H., Rizzo, G., Steinmetz, N., Mladenić, D., Auer, S., Lange, C. (eds.) ESWC 2016. LNCS, vol. 9989, pp. 46–49. Springer, Cham (2016). doi:10.1007/978-3-319-47602-5_10
5. De Meester, B., Maroy, W., Dimou, A., Verborgh, R., Mannens, E.: Declarative data transformations for linked data generation: the case of DBpedia. In: Blomqvist, E., Maynard, D., Gangemi, A., Hoekstra, R., Hitzler, P., Hartig, O. (eds.) ESWC 2017. LNCS, vol. 10250, pp. 33–48. Springer, Cham (2017). doi:10.1007/978-3-319-58451-5_3
6. Dimou, A., Kontokostas, D., Freudenberg, M., Verborgh, R., Lehmann, J., Mannens, E., Hellmann, S., Van de Walle, R.: Assessing and refining mappings to RDF to improve dataset quality. In: Arenas, M., et al. (eds.) ISWC 2015. LNCS, vol. 9367, pp. 133–149. Springer, Cham (2015). doi:10.1007/978-3-319-25010-6_8
7. 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: Proceedings of the 7th Workshop on Linked Data on the Web, CEUR Workshop Proceedings, vol. 1184 (2014)
8. Heyvaert, P., Dimou, A., Verborgh, R., Mannens, E.: Ontology-based data access mapping generation using data, schema, query, and mapping knowledge. In: Blomqvist, E., Maynard, D., Gangemi, A., Hoekstra, R., Hitzler, P., Hartig, O. (eds.) ESWC 2017. LNCS, vol. 10250, pp. 205–215. Springer, Cham (2017). doi:10.1007/978-3-319-58451-5_15

9. 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: Sack, H., Blomqvist, E., d'Aquin, M., Ghidini, C., Ponzetto, S.P., Lange, C. (eds.) ESWC 2016. LNCS, vol. 9678, pp. 709–723. Springer, Cham (2016). doi:10.1007/978-3-319-34129-3_43
10. Kontokostas, D., Westphal, P., Auer, S., Hellmann, S., Lehmann, J., Cornelissen, R., Zaveri, A.: Test-driven evaluation of linked data quality. In: Proceedings of the 23rd International Conference on World Wide Web, pp. 747–758. ACM (2014)
11. Lanthaler, M.: Hydra core vocabulary. Unofficial Draft, June 2014. http://www.hydra-cg.com/spec/latest/core/
12. Lehmann, J., Isele, R., Jakob, M., Jentzsch, A., Kontokostas, D., Mendes, P.N., Hellmann, S., Morsey, M., Van Kleef, P., Auer, S., Bizer, C.: DBpedia – a large-scale, multilingual knowledge base extracted from Wikipedia. Semant. Web (2015)
13. Paulheim, H.: Data-driven joint debugging of the DBpedia mappings and ontology. In: Blomqvist, E., Maynard, D., Gangemi, A., Hoekstra, R., Hitzler, P., Hartig, O. (eds.) ESWC 2017. LNCS, vol. 10249, pp. 404–418. Springer, Cham (2017). doi:10.1007/978-3-319-58068-5_25
14. Paulheim, H., Gangemi, A.: Serving DBpedia with DOLCE – more than just adding a cherry on top. In: Arenas, M., et al. (eds.) ISWC 2015. LNCS, vol. 9366, pp. 180–196. Springer, Cham (2015). doi:10.1007/978-3-319-25007-6_11
15. Regalia, B., Janowicz, K., Gao, S.: VOLT: a provenance-producing, transparent SPARQL proxy for the on-demand computation of linked data and its application to spatiotemporally dependent data. In: Sack, H., Blomqvist, E., d'Aquin, M., Ghidini, C., Ponzetto, S.P., Lange, C. (eds.) ESWC 2016. LNCS, vol. 9678, pp. 523–538. Springer, Cham (2016). doi:10.1007/978-3-319-34129-3_32
16. Schmachtenberg, M., Bizer, C., Paulheim, H.: Adoption of the linked data best practices in different topical domains. In: Mika, P., et al. (eds.) ISWC 2014. LNCS, vol. 8796, pp. 245–260. Springer, Cham (2014). doi:10.1007/978-3-319-11964-9_16
17. Wienand, D., Paulheim, H.: Detecting incorrect numerical data in DBpedia. In: Presutti, V., d'Amato, C., Gandon, F., d'Aquin, M., Staab, S., Tordai, A. (eds.) ESWC 2014. LNCS, vol. 8465, pp. 504–518. Springer, Cham (2014). doi:10.1007/978-3-319-07443-6_34
18. Zaveri, A., Kontokostas, D., Sherif, M.A., Bühmann, L., Morsey, M., Auer, S., Lehmann, J.: User-driven quality evaluation of DBpedia. In: Proceedings of the 9th International Conference on Semantic Systems, pp. 97–104. ACM (2013)