SAD Generator: eating our own dog food to generate KGs and websites for academic events

Pieter Heyvaert[1], David Chaves Fraga[2], Freddy Priyatna[2], Juan Sequeda[3], and Anastasia Dimou[1]

[1] IDLab, Dept of Electronics and Information Systems, Ghent University – imec
{pheyvaer.heyvaert,anastasia.dimou}@ugent.be
[2] Ontology Engineering Group, Universidad Politécnica de Madrid
{dchaves,fpriyatna}@fi.upm.es
[3] Capsenta
juan@capsenta.com

Abstract. Nowadays, a website is used to disseminate information about an event (e.g., location, dates, time). In the academic world, it is common to develop a website for an event, such as workshops or conferences. Aligning with the "Web of data", its dissemination should also happen by publishing the information of the event as a knowledge graph, e.g., via RDF that is available through a SPARQL endpoint or a Triple Patterns Fragment server. However, the RDF generation and website development is not always straightforward and can be time-consuming. In this demo, we present the Semantic Academic-event Dissemination (SAD) Generator for generating RDF and websites for academic events. The generator allows to (i) annotate CSV files that contain academic event data and use the annotations to generate a knowledge graph and (ii) generate a website with the information for the event querying the knowledge graph. We used our generator to generate the RDF and website of a real workshop, the KGB workshop. It can be easily reused by organizers of other academic events by simply providing the event's information in CSV files.

Keywords: RML · RDF · GraphQL · SPARQL · YARRRML

1 Introduction

In many domains, a website is one of the most frequent dissemination tools used to reach a wide range of audiences. Typically, a website is generated when an event is organized, containing useful information for potential attendees, such as the location, the dates or its main description. In the academic world, different kinds of event are usually organized, such as conferences, workshops, research schools or seminars where a corresponding website may be provided.

In the context of workshops and conferences, the information showed by their websites has usually the same structure. The website may show general information about the event (name, dates, location), call for papers together with corresponding topics, organizers, important dates and program committee.

This data can be easily defined using semi-structured data formats, such as CSV files or Excel files.

The resources of the event should also be published as a knowledge graph to be aligned with the "Web of data", following the Semantic Web/Linked Data approach: instances are identified by URLs and their types and properties are properly annotated. However, this step is usually seen as an additional feature for the dissemination, so it is avoided and, even if done, it remains unrelated to the website generation process. In fact, many of these websites obtain the data from isolated data sources where the used format is selected by the Web developer (e.g, databases, JSON or XML) and the data model is specific for each case. Recommendations, like RDFa, provide mechanisms to solve some of these issues, but its uptake is low, as it needs a manual intensive effort.

In this demo paper, we propose the SAD Generator[4], a Semantic Academic-event Dissemination Generator that uses Semantic Web technologies to generate a knowledge graph and website for academic events. This knowledge graph provides the data for generating the website. Furthermore, the knowledge graph can be made available via, e.g, a SPARQL endpoint or a Triple Pattern Fragments (TPF) server. We use this generator to build the knowledge graph and website of Knowledge Graph Building Workshop 2019[5] (co-located with ESWC2019).

2 The SAD Generator

When using our generator two main steps are executed: (i) the **knowledge graph** is built based on the raw data and, then, (ii) the **website** is generated based on this graph (see Figure 1). To achieve that, our generator considers the following data: (i) **CSV files** provided by the organizers of the event and describe the workshop itself, the organizers, programme committee (PC), topics, subtopics, and important dates; (ii) the **knowledge graph** that is generated based on the CSV files; and (iii) **HTML pages** of the website that are generated based on the knowledge graph. In the remaining, we discuss the two steps in more details: in Section 2.1, we discuss how the knowledge graph is built and, in Section 2.2, we elaborate on the website's generation.

2.1 Building the knowledge graph

In this section, we discuss the data model used for the knowledge graph, and the YARRRML rules [1], created based on the data model, that are executed to build the knowledge graph based on the CSV files.

Data model We annotate the different entities defined in the raw data semantically mostly using the Schema.org vocabulary to increase their reusability and sharability. A workshop is annotated as an event with the class `schema:Event`. The name, duration, conference co-located event, location, start date and end

[4] Available online at: `https://github.com/kgb-workshop/sad-generator`
[5] `http://kgb-workshop.org/`

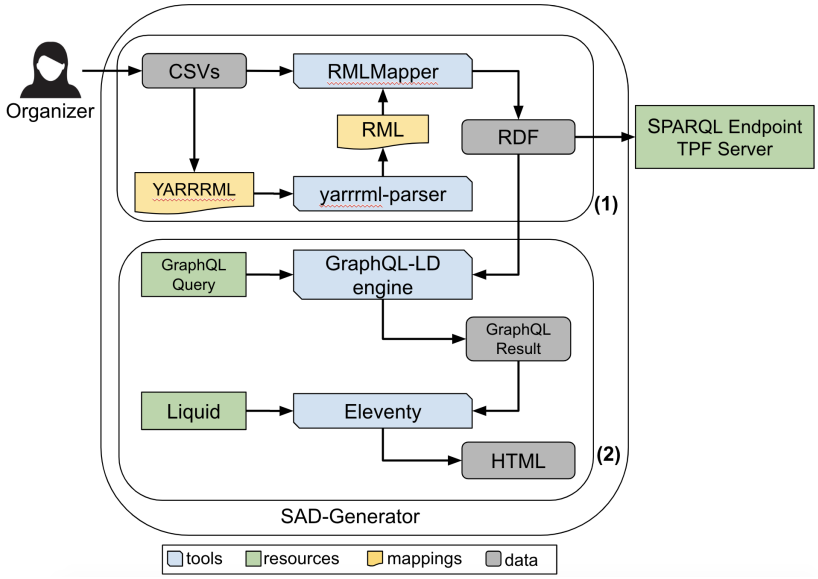

Fig. 1. SAD Generator workflow

date are added as datatype properties. The organizers, program committee, (sub)topics, and important dates are added as object properties. Organizers and PC members are annotated as a person with the class `schema:Person`. Important dates are annotated as events with the class `schema:Event` and for the sub(topics) we use the class `schema:CreativeWork`.

Listing 1.1. YARRRML rules for the PC members

```
sources:
  - ["csv/pc.csv~csv"]
s: https://kgb-workshop.org/resources/ProgramCommittee/$(id)
po:
  - [a, schema:Person]
  - [schema:name, $(name)]
  - [schema:memberOf, $(organization)]
  - [schema:performerIn,
    https://kgb-workshop.org/resources/Event/kgb2019~iri]
```

YARRRML rules Based on the structure of the CSV files and the data model, we use Matey[6] to define the corresponding YARRRML rules to build the knowledge graph[7]. These rules converted to RML rules [2] using the YARRRML Parser[8]. Initially, the RML rules are executed using the RMLMapper[9], which outputs the

[6] `http://rml.io/yarrrml/matey/`
[7] `https://github.com/kgb-workshop/sad-generator/blob/master/kg/mapping.yml`
[8] `https://github.com/rmlio/yarrrml-parser`
[9] `https://github.com/rmlio/rmlmapper-java`

first version of the knowledge graph as RDF. If the data is updated, we execute the RML rules again and a new knowledge graph is build. If the structure of the CSV files or the data model is updated, we update the YARRRML rules and convert them to RML rules, and execute the latter to build a new knowledge graph. The YARRML rules that define how to generate RDF describing the PC members can be found in Listing 1.1. They state that the relevant CSV file can be found in the folder `csv`, the URL of PC member is generated based on the `id` of a member, and that members are annotated with the class `schema:Person`, their name, organization and event they are performing in.

2.2 Generating the website

In this section, we discuss how we use an existing static site generator, Eleventy[10], to generate a website based on our knowledge graph: how the graph is queried and how the results of the queries are used to generate the HTML. We use a static site generator to reduce complexity, i.e., only a static file server is needed to host the website, and improve scalability, i.e., the website can be easily replicated.

We need to query the knowledge graph for the relevant data to be used in the HTML pages. We use a Comunica engine to execute GraphQL-LD queries on the RDF file. Comunica [3] is a highly modular and flexible meta query engine for the Web that allows executing queries across different RDF data sources, including RDF files, SPARQL endpoints, and TPF servers. GraphQL-LD [4] brings GraphQL to RDF: GraphQL-LD queries are translated to SPARQL queries. We use GraphQL-LD because of the Web developer-friendly approach of GraphQL and because the output of these queries are JSON arrays, which can then be used by template languages, such as Liquid[11], to build HTML pages. Comunica allows building a query engine that supports these GraphQL-LD queries on RDF files, which we use with Eleventy to query our knowledge graph[12].

Listing 1.2. GraphQL-LD query to retrieve the list of PC members

```
{ ... on Person
  {
    name
    affiliation
    event(_:workshop)
  }
}
```

Listing 1.3. Result of GraphQL-LD query

```
[ { name: "Amrapali Zaveri",
    affiliation:
      "Maastricht University"
  }, ...
]
```

Listing 1.4. Using the results in HTML

```
<ul>
  {% for person in programcommittee %}
```

[10] `https://11ty.io`
[11] `https://shopify.github.io/liquid/`
[12] `https://github.com/pheyvaer/graphqlld-on-file`

```
    <li>{{person.name}}, {{person.affiliation}}</li>
  {% endfor %}
</ul>
```

In Listing 1.2 the GraphQL-LD query that returns the PC members can be found. We only look for persons in our knowledge graph that have a name, affiliation, and that perform in a specific workshop. The URL of the workshop is assigned to `workshop`. The result is a JSON array containing an object for each PC member (see Listing 1.3). These objects are used in an HTML template with Liquid to list all the PC members (see Listing 1.4).

During the demo, we show how users can easily use the generator for their own events and how the changes to the CSV files result in an updated knowledge graph and website. Furthermore, we have already used this generator to build the knowledge graph[13] and the website[14] of Knowledge Graph Building Workshop 2019 (co-located with ESWC2019).

Acknowledgements

The described research activities were funded by Ghent University, imec, Flanders Innovation & Entrepreneurship (AIO), the Research Foundation – Flanders (FWO), and the European Union. The work presented in this paper is partially supported by the Spanish Ministerio de Economía, Industria y Competitividad and EU FEDER funds under the DATOS 4.0: RETOS Y SOLUCIONES - UPM Spanish national project (TIN2016-78011-C4-4-R) and by an FPI grant (BES-2017-082511).

References

[1] Pieter Heyvaert et al. "Declarative Rules for Linked Data Generation at Your Fingertips!" In: *European Semantic Web Conference*. Springer. 2018, pp. 213–217.

[2] Anastasia Dimou et al. "RML: A Generic Language for Integrated RDF Mappings of Heterogeneous Data". In: *LDOW*. 2014.

[3] Ruben Taelman et al. "Comunica: a modular SPARQL query engine for the web". In: *International Semantic Web Conference*. Springer. 2018, pp. 239–255.

[4] Ruben Taelman, Miel Vander Sande, and Ruben Verborgh. "GraphQL-LD: Linked Data Querying with GraphQL". In: *ISWC2018, the 17th International Semantic Web Conference*. 2018.

[13] `https://github.com/kgb-workshop/data`

[14] `https://github.com/kgb-workshop/website`