FAIRLibJS - Towards the fast and easy FAIR metadata annotation of serverless web applications

Martins, Yasmmin

yPublish

January 26, 2024
FAIRLibJS - Towards the fast and easy FAIR metadata annotation of serverless web applications
Yasmmin Martins¹

¹’yPublish,., Porto Alegre, Brazil

Corresponding authors:
Yasmmin Côrtes Martins, PhD.
yPublish
Porto Alegre, Brazil
e-mail: ycfrenchgirl2@gmail.com

Code availability
The code is available in a public repository at
https://epiverse.github.io/modules/fairlibjs
Abstract

Introduction: The FAIR principles stand for four important features that research objects such as software, methods and data should follow in order to allow other researchers to replicate the analysis and the results. The four components of these principles are the findability, accessibility, interoperability and reusability, and most the software available and associated to publications are findable by versioning systems such as github, but often they do not meet the other criteria, either because there are missing data, lack of documentation about the usage or testable examples to guarantee the appropriate execution. Regarding the usability, the metadata describing the objects allow the correct categorization of the research objects and proper recommendation of its application. But the metadata generation using semantic standards requires some knowledge of controlled vocabularies to describe them correctly, allowing an exclusive portion of the community to fully meet the FAIR principles.

Results: We implemented the FAIRLibJS that provides means to annotate Javascript software libraries allowing serverless web applications achieve the fourth FAIR criteria through correct metadata generation following semantic metadata standards. Besides the software metadata generation, the tool also enables the search and automatic semantic description of scientific publications indexed in OpenAire. The third main functionality is the extraction of keywords for any web application software and, for those directed to the biological domain, it also provides a mapping with the EDAM ontology topic and operation concepts. The JavaScript library and the Web module with usage examples and functionality demonstrations is provided to the public domain with open source.

Conclusion: The FAIRLibJS was able to extract correct keywords, from a given tool summary, for the manually annotated workflows in workflowHub, and also chose correctly the topics and operations concepts for more than 90% of the workflows. We also showed that through the usage of a few extra commentary syntax for documentation strings, the software developers are able to use our tool to derive enriched metadata to describe the Javascript modules, their functions with inputs and returned outputs and the module dependencies. The tool also provides a friendly interface to search data in OpenAire platform, enabling the automatic annotation of scientific publications.

Keywords: AI prompt engineering; FAIRrification, topic extraction, semantic annotation

Background

Recently, an increasing number of serverless applications has been observed to deal with research problems (Spillner, Mateos, and Monge 2018; Eismann et al. 2021). These applications have the advantage of not depending on installations in the user machine to perform their analysis and also offers privacy on analyzing data in a federated manner without exchanging or uploading data to a third party server. Although they are in compliance with many of the FAIR principle rules (Lampecht et al. 2020; Wilkinson et al. 2016) by being public accessible, findable, interoperable and reusable, most of them fail to present the metadata concerning basic information about the involved libraries and tools.

There are great initiatives that promote the generation of metadata to describe research data and software, such as the Elixir (Harrow et al. 2021), and this metadata can be uploaded by the research product authors into registries such as bio.tools (Ison et al. 2019) and workflowhub (da Silva et al. 2020). The workflow hub is directed to computational workflows.
and currently do not accept descriptions of javascript applications. The workflow metadata is entered manually while in bio.tools there is a working API to recover and upload the metadata about the tools automatically. Bio.tools supports javascript libraries and both registries enable the assignment of the enriched descriptors involving EDAM concepts. However, to use the automatic api to upload the metadata, the research product author needs a minimum knowledge to produce the json with the correct properties and classes, following the Bioschemas (Castro et al. 2023) or other controlled vocabularies and ontologies (Staab and Studer 2010). A small portion of research product authors also has this expertise which makes it difficult for the adoption of the metadata generation to be in compliance with the FAIR standards.

The Research Object Crate (Creators Ó Carragáin, Eoghan1 Goble, Carole2 Sefton, Peter3 Soiland-Reyes, Stian2 Show affiliations 1. University College Cork 2. The University of Manchester 3. University of Technology Sydney, n.d.) provides guidelines to annotate research products such as datasets, publications, analytical methods, software, etc in a machine-readable fashion, using the JSON-LD structured data and applying widely used controlled vocabularies such as schema.org (Meusel, Bizer, and Paulheim 2015) and dublin core (Weibel and Koch 2000). Some software packages emerged to help scientists annotate their research products programmatically, and there exist a library¹ (RO-crate-js) specifically designed for Javascript that helps the generation and packing of the descriptions along the semantic graph. However, besides it abstracts the creation of nodes in comprehensible methods, the person still needs to have some knowledge level about the relations to describe the objects correctly and the library does not support data retrieval and enrichment from other databases such as OpenAire (Manghi et al. 2019) that collects and indexes published research products.

Considering these points, we present the FAIRLibJs that enables three main functionalities concerning the FAIR manner to share research resources following recommended standards: (i) the generation of a machine-readable structured JSON-LD from human friendly commentaries extended from software documentation best practices; (ii) data retrieval and parsing for the OpenAire access programming interface (API), with independent enrichment and JSON-LD metadata resolution for the scientific publications; and (iii) automatic extraction of candidate keywords from a text summary and mapping to EDAM topic and operation concepts.

https://content.iospress.com/articles/data-science/ds190026

¹ https://www.npmjs.com/package/ro-crate
Material and methods

FAIRLibJS architecture

FAIRLibJS library is organized according to three main aspects of web software applications semantic metadata annotation: (i) the automatic generation of a JSON-LD representing the JavaScript libraries; (ii) the extraction of topics or keywords to annotate the tools and, specifically for bioinformatic tools, the alignment with EDAM ontology; (iii) the automatic data retrieval from OpenAire and semantic metadata annotation and enrichment of scientific publications.

Software Library Annotation

The proper code documentation explaining the minimum information of each function of a module is essential to allow the maintainability, extension, comprehension, readability, and mainly, reusability by the community. Therefore, we followed documentation best practices and created a light-weight version (GenDoc) of JsDoc² to parse commentary blocks with the correct markup keywords that renders in a given container identifier the HTML with the objects described in a given source code and function information. We also extended the markup keywords to add information concerning authorship, provenance and requirements. GenDoc is an auxiliary library to present information about the library objects, name, description, list of properties and specific details about the properties, such as name, description, namespace, parameter details and usage example as command lines. These parameter details are the variable type, name, default value (if exists) and an optional description. The FAIRLibJs aggregates and parses other information such as dependencies, author information, license, unique identifiers and the granularity. The granularity markup allows the identification of full libraries (collection of functions) and the individual functions. The linkage among the functions and the library is made using the markup @meta.belongsTo that expects an URI corresponding to the library identifier previously assigned. The authorship markup accepts properties like identifier, name and orcid³ (external identifier used by academics and scientists). The license markup value, discriminating all the allowed usage scope of the software, accepts any of the identifiers listed in the SPDX website⁴, such as MIT, BSD-3-Clause-Flex, CC-BY-2.0. The markup to represent dependencies is similar to the markups accepted for the library and functions, such as the URI, type (function or module), name, the author URI and name. These dependencies and the authorship are an important information source to track provenance (Sahoo, Valdez, and Rueschman 2016; Gil et al. 2016) and we allow the dependency specification of a full library or a specific functionality of external or internal libraries. We also describe the input parameters and function outputs to track provenance from the data generation point of view, reusing the GenDoc functionality to parse the parameters and returned variables. Finally, we also allow the addition of alternative resources being described as reference markup, that will be annotated with the property seeAlso from the RDF (Resource Description Framework) Schema vocabulary (K. K. Breitman, Casanova, and Truszkowski 2007).

---

² https://jsdoc.app/
³ https://orcid.org/
⁴ https://spdx.org/licenses/
Most of the properties used to annotate comes from the bioschemas vocabulary (Castro et al. 2023); the alternative labels and external references to the resources use the RDF Schema; the dublin core vocabulary is used to add the conformsTo property to attach the correct Bioschemas profile for the resources; the edam ontology (Ison et al. 2013) was used to assign the topic and operations but also to map the json (object - edam:format_3464) and Array (edam:data_2082) data types found in parameters and output details. The other primitive data types were mapped and assigned with the XML Schema vocabulary (Vlist 2002). The FAIRLibJs source code\(^5\) contains examples of all these descriptors, both for documentation and JSON-LD data generation.

### Topic extraction and annotation

This functionality aims at extracting keywords from a summary describing a research software and then map these keywords to the topics and operations described in the EDAM ontology. Currently, these keywords (tags) are filled as a form manually while registering a tool in the bio.tools (Ison et al. 2019) and workflowhub (da Silva et al. 2020) registries. Although these fields are not required to register new applications, the correct assignment of tags helps the researchers find the resources efficiently according to their application domain.

The summary text is cleaned and transformed before the topic extraction, since the person may pass directly the markdown content of a readme file from their github repository or an HTML markup content, the function detects the non-desired characters and tags and saves only the raw text.

This functionality offers two methods to extract the most relevant topics from the transformed summary text, the usage of the chatgpt (Biswa 2023) ith a prompt specifically designed for the topic extraction containing the content “You are a helpful assistant that extract the most significant keywords from a text with at least two tokens, show them separated by comma” passed as the assistant role along with the summary text in the user role. The model version used is the “gpt-3.5-turbo” with temperature 0.7, the function following this settings uses the Open AI API\(^6\) to get the topic candidates.

This version of the topic extraction needs a valid open AI key to send the request correctly and it has a certain cost according to the size of the text and the volume of requests. To overcome this issue, we also implemented a completely free function that uses the rapid automatic keyword extraction (rake) algorithm (Baruni and Sathiaseelan 2020) that analyzes the text locally and returns the most relevant sets of terms by calculating the frequency of the word groups and weighting according to the co-occurrence with other words in the text. Before applying the rake method, we treat the text removing the stop words that we define as all that do not belong to the noun and adjective classes, using a part of speech tagger algorithm (Brill 1992).

From the extracted keywords, our library offers the alignment of these term sets against the EDAM ontology’s topics and operations to annotate these keywords. While instantiating the main object library, it imports the necessary libraries and also calls an internal utility function to extract the URIs, names (labels) and synonyms of the non deprecated classes that were child of the topic and operation root classes, identified by the “#operation_0004” and

---

\(^5\) https://github.com/epiverse/modules/tree/main/fairlibjs

\(^6\) https://openai.com/blog/openai-api
“#topic_0003” respectively. To map the extracted keywords to these concepts, we selected a committee formed by a set of four string similarity metrics: Cosine (Rahutomo, Kitasuka, and Aritsugi 2012), Jaro Winkler (Wang, Qin, and Wang 2017), Levenshtein (Yujiian and Bo 2007) and Tri-Gram (Islam, Milios, and Kešelj 2012). The Cosine and Jaro Winkler values are already given as similarity index, but the Levenshtein is returned as distance and the value was transformed in similarity by extracting the ratio in relation to the keyword length and subtracting one. The trigram returns the an inverse sorted list of candidates and the number of matches found for the tri character sets of the pair in comparison. These matches were divided by the length of the term to return the similarity value in the same range (0 to 1) as the other metrics. The comparison pairs are generated among the keywords and the labels and synonyms of the EDAM concepts, and the final similarity for each metric is given by the maximum similarity found among these synonyms. The mapping function returns only the top 10 concepts, from all that were tested against the keywords, that obtained a mean similarity of at least 0.6 from the committee.

OpenAire data retrieval and Articles annotation

In order to complement the automatic semantic description of research products, we also provide data retrieval functions to query each type of data available in the OpenAire database (Manghi et al. 2019) calling their exposed API endpoint7. The functions receive a dictionary containing the valid parameters for each type of research product or project, validates the values, encodes and sends the request, parsing the XML returned with the hits or treating the possible errors. The OpenAire graph contains data about public projects, our research products, among these three are specifications such as data, software, publication or other types of products. The most common asset found are the publications, so we enriched the default object parsed from the results in XML adding descriptors from the dublin core ontology concerning all the data available for the publications and the provenance with the creator and publisher nodes. In general, the available attributes are the title, abstract (description), publisher, date of acceptance and the authors. The list of authors are given by resources of type Person aggregated in an array in each publication as the value of the property authorList from the bibliographic ontology (bibo namespace). The library has also a helper utility function to export the produced JSON-LD enriched files for download. Then, they can be embedded in HTML pages and correctly crawled (Navarrete et al. 2019).

Case study and strategies evaluation

Topic extraction and annotation

To evaluate the correct keywords extraction according to the summary text of the software being described, we retrieved the curated annotated metadata from workflows hosted in the Workflow Hub registry up to October, 26. We scraped and parsed the html pages containing the list of workflows to retrieve their identifiers and the respective name and authors. Then, the individual pages were scraped to obtain the detailed annotations (Khder 2021). We were mainly interested in the keywords, EDAM topics and operations, besides the workflow pages has a json-ld attached in a script tag, only the annotated keywords are contained in it. The topics and operations selected by the workflow creator were mined along the HTML anchor

7 https://graph.openaire.eu/develop/api.html#projects
tags that included the edam browser link and extracted the id from the end of the link in the href property while the main label was returned from the inner text. The parsed information were saved as a json file to be easily parsed in our web application to compare and evaluate those predicted with the real chosen keywords and EDAM concepts.

We build a function to perform the mapping and extraction of keywords in batch, from a list of objects with at least the “description” property. In case the person wants to check and evaluate with real annotated keywords and concepts, there must be a property keywords with a list of strings, and “topics” and/or “operations” properties that are expected to be objects with id (EDAM ontology identifier) and name (label). There is a helper function to evaluate two lists of the same type according to a cutoff value, the main batch function in evaluation mode uses this helper one for the keywords, topics and operations separately.

The evaluation function measures for each predicted term the max similarity found in the true list of terms, and assigns a binary class 1 in case the max similarity found by at least one of the metrics in the committee is higher than the passed cutoff value. We used the accuracy score to measure the number of successful predicted terms in relation to all the ones that were extracted and mapped.

Library’s metadata generation

We evaluated the metadata generated and exported in JSON-LD by annotating and embedding the semantic graph from our own library source code commentaries. This enriched json file was put on the web page illustrating the three main functionalities and we submitted to the FAIR-Checker tool (Gaignard et al. 2023) to examine the quality of the metadata according to the four dimensions of the FAIR principles (Findable, Accessible, Interoperable and Reusable).

This tool has an api as well as a web application that has two main functions: inspect and check. We used only the check function that given a URL that has some kind of RDFa, microdata, JSON-LD or metadata markups, it tries to identify standard properties and classes concerning accessibility, license, provenance in relation to the authors, creators and contributors, the usage of ontology terms registered in broadly known catalogs (Vandenbussche and Atemezing 2017; Noy et al. 2009; Vrousgou et al. 2016), etc. The checking function attributes scores from 0 to 2 for specific constraints in each of the four dimensions, the higher the score the more in accordance with the FAIRification cookbook.

Since we added all the complete information about the libraries and the functions, using all the markups proposed, we expected to pass successfully by all the metrics.

Results

Topic extraction and annotation

The data scraped from the Workflow Hub registry comprised 410 workflows submitted up to October 26. On uploading a workflow into the registry, the user may annotate tags (keywords) and also apply topics and/or operations described in the EDAM ontology. From these, the metadata exploration indicates that only 88 were annotated with at least the

---

8 https://epiverse.github.io/modules/fairlibjs/index.html
9 https://faircookbook.elixir-europe.org/content/search-wizard.html
EDAM topic concepts, while all those records that have keywords (64) also had the operation concepts present. Interestingly, when the workflow authors use to annotate keywords they seem to add with EDAM topics and also the operations, while the topics were the only annotation for 24 workflows. The average number of keywords chosen per workflow is close to 5, while only about 2 EDAM concepts are chosen both for topic and operation branches. We also observed that considering all the 410 workflows the mean number of characters used to describe and summarize the workflow is around 2300, but in the subgroup of 88 fully annotated this number increases to 3200, suggesting that those that take a time to curate tags and concepts are willing to provide more details about the workflows. After cleaning the HTML and Markdown markups, the average description texts had approximately 2800 characters, this reduction avoids spending time processing unnecessary text.

We measured the execution time performance with the following granularities: the overall value spent to execute for all the 82 eligible workflows in all steps (keywords extraction and mapping of topics and operations); the keyword extraction according to the available methods (gpt or rake); the topics mapping and, when present, the operations mapping. The other evaluation criteria was the ability to find matchings among the predicted items and those curated by the workflow authors.

According to the prediction time comparison (Figure 1), it is clear that requests to the open ai to extract the keywords according to the designed prompt makes the time increase considerably. Besides the mapping step for rake takes more time than the one for gpt, the overall time for rake (31 minutes) is still reduced by 50% in relation to gpt (1h e 3 minutes). The rake method can be adjusted according to the stop words, token filtering (by part of speech) and score cutoff while the gpt does not provide this configuration flexibility.

![Prediction Time Comparison](image)

**Figure 1.** Comparison of the total execution time for each keyword extraction method (rake or gpt) stratified by the topic extraction, edam concepts mapping (topics and operations) and the overall time.

Figure 2 shows that the rake extraction generates more keyword candidates than the gpt, the difference directly influences processing time while calculating and mapping with the
topics and operations branches of the EDAM ontology. Although the mean number of curated keywords is about 4, some workflows were annotated with 20 keywords. The rake method offers a free option to get relevant topics contained in the summary scope, besides most of the candidates generated by gpt are reasonable, this technology tends to hallucinate and predict terms out of context (Alkaissi and McFarlane 2023).

**Figure 2.** Number of predicted keywords, topics and operations by topic extraction method (rake and gpt).

Since the structure of the terms predicted for the keywords may vary a lot, we evaluated the exact matchings only in relation to the topics and operations that are standard and have unique identifiers. The coverage of mapped items from the keywords was similar for both methods (Table 1), the RAKE method obtained a small advantage achieving more than 95%, which was expected because of the diversity of keyword options.

**Table 1.** Coverage of workflows in which the keywords found by each extraction method mapped with valid topic and operation terms. From the 82 workflows with some annotation, only 63 had at least one operation concept.

<table>
<thead>
<tr>
<th>(Coverage of Mapping) EDAM branch / Method</th>
<th>GPT</th>
<th>RAKE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Topic</td>
<td>76/82 (92.7%)</td>
<td>78/82 (95.12%)</td>
</tr>
<tr>
<td>Operation</td>
<td>59/63 (93.65%)</td>
<td>60/63 (95.23%)</td>
</tr>
</tbody>
</table>

In relation to the exact matches, both extraction methods yielded almost equal results of workflows that the predicted operation and topic matched the original annotations. For both cases, the maximum of terms in the intersection were 3. As shown in table 2, the coverage of workflows in which the intersection was not empty was up to two units higher for the GPT method. Although most of the coverage values were under 50%, using the similarity metrics
committee to reach those that are similar, we increased this percentage to a range of 85% to 92%. These results demonstrate that our tool is able to recommend correctly the concepts to annotate the tool from the summary.

Table 2. Coverage of workflows in which the predicted topics and operation concepts from the keywords were exactly the same as those chosen by the workflows’ authors.

<table>
<thead>
<tr>
<th>(Coverage of Exact Matches) EDAM branch / Method</th>
<th>GPT</th>
<th>RAKE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Topic</td>
<td>34/82 (41.46%)</td>
<td>33/82 (40.24%)</td>
</tr>
<tr>
<td>Operation</td>
<td>33/63 (52.38%)</td>
<td>31/63 (49.21%)</td>
</tr>
</tbody>
</table>

FAIR checking of library’s metadata

We evaluated the library metadata according to the documentation markups that we filled in our own source code, embedded in the github pages of the tool and submitted the url to the FAIR-checker API (Gaignard et al. 2023). As we expected, we obtained the highest scores for each of the four dimensions in all detailed criteria (Supplementary table in https://drive.google.com/file/d/1siNgbre-K9HcHA0-u4dPkmLGsEJ7sTVF/view?usp=sharing). One of the most important criteria is the provenance statement, and not only we stated the correct provenance for the library itself but also we described the authors of the libraries that are required by our tool. So, our descriptors are sufficient to generate a metadata that indeed covers all the minimum and recommended criteria for the FAIR dimensions. Our tool not only provides computational tool annotations but also adds functionalities to annotate scientific publications with all the pertinent descriptors and makes the annotation process easier from the user’s perspective. We also contributed with an accessory library (GenDoc) to generate and render in HTML the documentation concerning the library, its objects and properties. The libraries are independent from each other separating the documentation from the FAIR metadata generation. Another advantage of the FAIRLibJs and GenDoc is that they do not need any installation in the Node server side, that can be loaded and used instantly.

Conclusion

Generating semantic metadata to describe research products such as publications and software may be tricky and also requires a certain knowledge of ontologies and their catalogs to choose the classes and properties correctly. In this paper, we propose the FAIRLibJS whose main functionality is the annotation of Javascript modules proposing human-friendly markup keywords that are naturally added as documentation in commentary blocks. As satellite contributions we provide an interface to query and parsing results of the OpenAire API, specially generating a JSON-LD in the publications result. The third main functionality is the keywords extraction from the tool summary and recommendation of topics and operations from EDAM ontology. We evaluated the EDAM concepts mapping by comparing our predictions with real annotated key words, topics and operations from the

---

10 https://github.com/epiverse/modules/blob/main/gendoc.js
workflows hosted in workflowhub catalog. While the produced library metadata was analyzed regarding the annotation criteria for Findable, Accessible, Interoperable and Reusable principles, using the tool FAIR-checker. The results demonstrated that our tool is able to correctly find relevant keywords according to a strict similarity metrics committee and the metadata automatically built from the markups achieved the highest score in all criteria evaluators. In fact, our markups outperforms in the sense of annotating the dependencies and correctly annotating the library components, and the input and output of all the functions under the library scope.


formats." Bioinformatics 29 (10): 1325–32.
WIMS '15 15. New York, NY, USA: Association for Computing Machinery.