Abdel-Qader, Mohammad; Saleh, Ahmed; Tochtermann, Klaus

Book Chapter — Published Version

The Technical Specifications and Requirements for Connecting OER Repositories Using the LOM Standard

Suggested Citation: Abdel-Qader, Mohammad; Saleh, Ahmed; Tochtermann, Klaus (2023) : The Technical Specifications and Requirements for Connecting OER Repositories Using the LOM Standard, In: Otto, Daniel et al. (Ed.): Distributed Learning Ecosystems, Springer VS, Wiesbaden, pp. 227-240, https://doi.org/10.1007/978-3-658-38703-7_12

This Version is available at:
http://hdl.handle.net/11108/644

The Technical Specifications and Requirements for Connecting OER Repositories Using the LOM Standard

Mohammad Abdel-Qader, Ahmed Saleh and Klaus Tochtermann

Abstract

One of the goals of creating Open Educational Resources (OER) is to increase their accessibility for more learners. Connecting the different repositories that provide that these OER use one standard can help achieve that goal. In this chapter, we give detailed specifications and requirements for connecting different OER repositories using the Learning Object Metadata (LOM) standard from a technical point of view. We define the used technical terms and show how the process is working at the back end. More specifically, for each stage of connecting repositories, starting from harvesting the metadata from those repositories to storing the processed data in files ready to be used in the front end, we describe the functional requirements, what technologies are needed, and how the process works. In this chapter, we will describe the process of connecting the OER repositories using the LOM standard from start to end as simply as possible. The idea is to allow non-technical staff to replicate such a process, or maybe some stages of it. Afterwards, we give some examples of the tools that may help in the process of harvesting data from the web. Some

M. Abdel-Qader (✉) · A. Saleh · K. Tochtermann
ZBW—Leibniz Information Centre for Economics, Kiel, Germany
e-mail: m.abdel-qader@zbw.eu

A. Saleh
e-mail: a.saleh@zbw.eu

K. Tochtermann
Christian-Albrecht University of Kiel, Kiel, Germany
e-mail: k.tochtermann@zbw.eu

D. Otto et al. (eds.), *Distributed Learning Ecosystems*,
https://doi.org/10.1007/978-3-658-38703-7_12

of these tools are visual and do not require any programming skills. Finally, we briefly describe the EduArc project, which connects OER repositories using the LOM standard.

1 Introduction

There are many providers of educational resources, such as educational institutes and universities. One form of educational resource is the Open Educational Resources (OER). OER can be defined as resources used for learning and published under the license of open access (Hylén, 2006), which can be provided in many formats, such as videos, slides, etc. The metadata of the OER is the data that describes the OER and is stored in a database management system or index.

The OER providers have many options to model their metadata. One option is to use the existing standards such as the Learning Object Metadata (LOM) standard (IEEE 2002), the Learning Resource Metadata Initiative (LRMI),[1] and the Metadata Object Description Schema (MODS).[2] The second option is not to use any of the existing standards and to model their own style.

Connecting the OER repositories using one standard can increase the accessibility of these OER resources. Furthermore, it can achieve one of the principles of FAIR as referenced by Wilkinson (Wilkinson et al., 2016), which is Interoperability. In general, FAIR principles represent data publishers' guidelines to providing their data using digital publishing with maximum possible added value. The idea of the interoperability principle is that the data must work in conjunction with other data and applications. Additionally, using one modeling standard to represent the OER will facilitate sharing the metadata among the OER providers.

In order to connect these different OER repositories using one standard, we follow three steps. Figure 1 shows these stages on a conceptual level. More details will follow below. The first stage shown in Fig. 1 is collecting or extracting data from the different OER repositories. This process is called harvesting. Afterwards, the harvested data will go into different processing steps. These steps include cleaning the harvested data and assigning to the data into the proper field of the LOM standard. The final stage is storing the resulting data. The processed

[1] https://www.dublincore.org/specifications/lrmi/1.1/, last accessed: October 10, 2021.

[2] https://www.loc.gov/standards/mods/, last accessed: October 10, 2021.

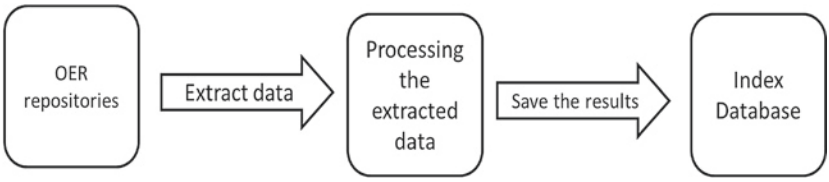

Fig. 1 The different stages for connecting OER repositories using known standards

data will be in the form of structured data, which can be stored in a database or in an index.

The detailed process is shown in Fig. 2. The process shows three scenarios (Abdel-Qader et al., 2021). The first scenario is when the OER repositories use the LOM standard to model their metadata. The scenario shows that when the data is harvested, it is immediately ready to be stored. This scenario is the best-case scenario as it follows all four FAIR principles: Findability, Accessibility, Interoperability, and the Reuse of digital resources. All you need is to obtain the data and then store it.

The second scenario shows that when the OER providers use any other standard but LOM to represent their metadata, a mapper must map the harvested data into the

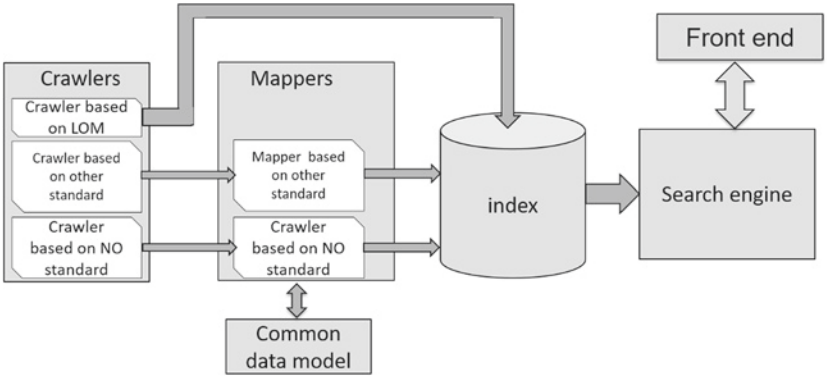

Fig. 2 The scenarios of harvesting and mapping the OER metadata that are modelled using the LOM standard, other standards, or no standard, starting from harvesting the metadata to storing the results and presenting them on the front end

LOM standard. In this case, we need a mapper for each standard. Then, each repository needs a dedicated mapper. The last scenario **occurs** when none of the existing standards is used. In that case, more work is needed from the developers. This last scenario would mean not applying FAIR principles since not using a standard will not achieve the accessibility, interoperability, and reusability principles.

When the data is mapped, it is ready to be stored in an index. Afterwards, a search engine can be developed to interact with the index. This will allow searching for open educational resources. A front end will facilitate that interaction with the index.

The remainder of this chapter is structured as follows. In Sect. 2, we describe the LOM standard and its main features and elements. The web harvesting process and the harvesting policies are described in Sect. 3. In Sect. 4, we show the process of metadata mapping. Sections 5 and 6 describe the results and how to store them. A brief description of some of the harvesting tools is given in Sect. 7. Finally, an example of one of the projects that connect OER using the LOM standard is shown in Sect. 8 before we summarize the main points.

2 The LOM Standard

The LOM standard (IEEE, 2002) is a data model to represent the metadata of educational resources, such as video lectures, presentation slides, or other formats. The LOM standard consists of a set of fields that specifies the format in which the metadata of the educational resources is stored. It controls the stored metadata to make sure that all the data follows the same rules and formats. The metadata is stored in a digital format. This allows the sharing and reusability of the metadata among different educational platforms.

The LOM standard consists of 15 main elements. These elements represent the structure of the metadata of the educational resources. Most of these 15 elements contain more detailed sub-elements in order to add more levels of information to describe the educational resources. A list of theses 15 elements is shown in Table 1.

As an example of these 15 elements, the LOM standard has the element “general”, which gives the most general information on the educational resource, such as the title, language, and keywords. Another example is the “technical” element. This element describes the technical properties and requirements of the learning resource. It consists of five sub-elements, such as the format, size, and duration information.

Table 1 The main 15 elements of the LOM standard

Metadata	Technical	Description
Lom	Educational	Datatime
General	Rights	Entry
Lifecycle	Classification	*{involved people}*
Meta-metadata	Langstring	*{Controlled vocabulary}*

3 Web Harvesting

The process of collecting and scraping the data off the web pages is called web harvesting. This includes the data shown on the web page and the data that is hidden and/or not shown. The web harvesting process is usually done by using computer programs (software) (Olston & Najork, 2010). These programs are generally known as web spiders or web robots. The harvested data can be stored in any format, such as in database form, JSON records, or simply in a sheet. Using web harvesting, we can crawl any amount of data, from simple web pages to a massive repository of web pages or resources.

In this section, we describe the harvesting policies the user needs to follow in order to collect data from the web in an efficient manner. Also, we describe the two types or techniques of harvesting, namely general harvesting and focused harvesting.

3.1 Harvesting Policies

Before harvesting any data on the web, one must have the right to collect this data. The rights of the data is owned by their authors/creators. Therefore, the copyrights should be checked before harvesting, or one should contact the authors or owners to obtain their permission to crawl the data from their portals. There are many copyright licenses available, and each one has different rules and specifications. If the copyright policy permits harvesting the data, one needs to follow the harvesting policies to collect the desired data from these web pages. There are four main web harvesting policies: the selection policy, the revisit policy, the politeness policy, and the parallelization policy (Castillo, 2005). The description of these policies is as follows.

- **The selection policy**: The size of the web is huge. Nowadays, there is a massive amount of web pages, and each web page can contain a large amount of data. Thus, the entire web cannot be harvested. In order to harvest some data from the web, one needs to set up a target by specifying the required type of data, the amount of data, and the number of web pages that contain this data.
- **The revisit policy**: After crawling a set of web pages that have been specified based on the previous selection policy, the content of these pages may change due to the dynamic behavior of the web. Therefore, in order to keep the harvested data up to date, one needs to revisit the already harvested web pages to crawl the updates and then update the database of the harvested data.
- **The politeness policy**: The web pages are hosted on servers, and these servers have resources, such as the memory and the bandwidth, which are limited. The process of harvesting a web page includes downloading that web page. The download will require the server resource to make your download request. So, the larger the harvested web pages, the larger the required resources. It is best to remember that many other users use or visit the web pages you are harvesting. Therefore, one should not overload the servers with requests to harvest many web pages simultaneously and, thus, prevent other users from using these web pages. It would be best to harvest data from the web politely. One can add pauses between sequences of requests so that the resources of the harvested resources are not monopolized.
- **The parallelization policy**: Some crawling software offers parallel crawling. This means that the user of the crawling software can run many crawlers at the same time. Using this approach, the download rate will be maximized, and the overload rate will be minimized as much as possible. The parallelization policy states that when the parallel crawlers are used, one needs to make sure that these parallel crawlers do not visit the same page severally. The web pages ought to be visited by only one crawler each. This will avoid wasting the resources of the servers and maximize the download rate.

3.2 General Harvesting

The general harvester is designed to crawl data from any web page without the need to modify the specification or the design of the harvester. The only changed item is the URL of the web page (Olston & Najork, 2010). This type of harvester has the main advantage that it is developed once and then used for crawling any web page. Despite this main advantage, it is limited regarding the amount of data

that can be harvested since each web page has a different structure, which makes tracking the data on the page more difficult.

3.3 Focused Harvesting

Due to the limitation on the amount of data crawled using the general harvesters, the developers can design a focused harvester designed to crawl data based on a specific topic or portal (Johnson et al., 2003). The developer analyses the structure of the desired portal or repository they want to collect data from, then design and develop a harvester that follows the structure of that portal to extract the required information.

The main advantage of the focused harvester is that it maximizes the amount of harvested data since the web page structure is analysed and the position of the data known so that it can be easily harvested. Furthermore, some website developers use the existing templates when designing their portals. From that, we can take advantage of these templates and develop harvesters based on the structure of the template that some of the web pages use. The main disadvantage of the focused harvester is that the developer needs to design a crawler for each repository or portal. This will lead to more work designing different crawlers for different portals, especially when the data that needs to be collected comes from many repositories. Thus, it is time-consuming for the developers of the crawlers.

4 Metadata Mapping

The harvested data from the web pages will be stored using the following pattern:

```
<field name> : <value>
```

`<field name>` is the name of the information that will be stored in your database after harvesting it from the web page, such as “title”, “abstract”, or “keywords”. You can name these fields as you like or in some cases, the designers of the web page name these fields. Thus, the field names will be harvested with the data. The `<value>` part is the actual information of the harvested field.

As mentioned above, the LOM standard has many elements, and these elements contain the fields’ names. In most cases, the names of these fields in LOM are different from those used to represent the data in the portals. For example, some could

name the field that represents the title of a lecture as "*titel*" (in German), and the LOM standard has a field name that represents the same information called "*title*". Therefore, we need to change the names of the fields of the harvested data to match the field name of the LOM standard. This process is called mapping, which is the translation of the harvested field name to another field name (Latif et al., 2021).

4.1 General Mapping

As in the general harvester, the general mapper can be used to map the fields from many repositories into the fields of the LOM standard without changing anything in the design of the mapper. Since the fields' names by using the general harvesters will be the same for all repositories, the general mapper will use these fields and find their matching field in the LOM standard. The main advantage is the same as the advantage of the general harvester; one mapper is developed for many repositories.

4.2 Focused Mapping

Since each data provider can use any of the available standards to represent their data or use their own representation structure, we require a common model for all the harvested data. To store the harvested data using one standard, which in our case is the LOM standard, we require all the information to follow the same structure in terms of the hierarchy of the data and the field names that represent the actual data.

Each focused harvester needs a mapper to match the field names in the LOM standard. These types of mappers are called focused mappers since each focused harvester needs a mapper. The main disadvantage of this type of mapper is that it is time-consuming for developers since they need to develop a mapper for each repository. This problem will occur especially if the number of focused harvesters is big.

5 Results

After the mapping stage, the processed data can be stored in different formats. One of these formats is the JavaScript Object Notation (JSON). The JSON format is used to store data that can be parsed by computers. This format is characterized by being human-readable and language-independent (Nurseitov et al., 2009).

```
{
        "title" : "How to store data using the JSON format. The easy guide",
        "author": {
                        "firstName" : "John",
                        "lastName" : "Smith"
                  }
        "publicationDate" : "01.01.2021",
        "abstract" : "In this article, we describe the process of storing data using
the Javascript
                        Object Notation (JSON) format. The JSON format is human
readable…"
}
```

Fig. 3 Example of a record that describes the information of an article using the JSON format

Another benefit of the JSON format is that the data can easily be stored, processed, and exchanged between different repositories. This becomes obvious when sharing the data of resources that are classified as open resources, such as the Open Educational Resources (OER).

The JSON format follows the pattern:

```
<field name> : <value>
```

This pattern can have any number of subfields to add more complexity and structure to the stored data. The files that will store the data using the JSON format will get a.json extension. Figure 3 shows an example of a JSON record that describes an article.

6 Storing the Results

After harvesting and mapping the crawled data from the web pages, the processed data is stored. One can use any storing method to save the processed data. It can be stored using JSON files with a.json extension, in a rational database management system, or in an index.

An index is a method to store a collection of documents to facilitate the search process. The index can be treated as a table in a rational database management system (Divya & Goyal, 2013). An example of an indexing system is Elastic-

search, which is a search and analytic engine for all types of data.[3] The main characteristic of Elasticsearch is that you can search the index almost in real-time. The Elasticsearch index can contain mapping rules that control the fields and the data that will be stored inside the index.

7 Harvesting Tools

There are several ways to harvest data from the web. Some need programming skills, while others do not require any knowledge in programming and software development. In this section, we explore different tools and libraries that are commonly used for harvesting data from web pages.

7.1 Harvesting Using Visual Tools

Data can be harvested using the available visual tools. The user of such tools does not need programming knowledge. Most of these visual harvesting tools are web-based, therefore, installing the tool on a device is not required. Despite the previous advantage, one of the limitations of the visual tools is that the harvested data requires more scrubbing. The harvested data is not clean and needs more work after the harvesting process is completed. Therefore, more work for the user starts after harvesting the data from the web. Another disadvantage is that most of these tools are not open-source, and you need to pay for the license. Below are some of the most common visual tools used and a brief description of each of them.

- Apify[4]: A web harvesting platform that downloads data in a structured form. It has some ready harvesters for some of the well-known data sources, such as Google Maps, Facebook, and Twitter. Apify has a free trial plan for 30 days.
- Import.io[5]: Usually, large companies use this tool. This tool is easy to use, and no programming skills are required. Yet, the main disadvantage is that this tool

[3] https://www.elastic.co/, last accessed: October 10, 2021.

[4] https://apify.com/, last accessed: October 10, 2021.

[5] https://www.import.io/, last accessed: October 10, 2021.

has to be run by the enterprise by itself with minimal support from the developers. Import.io is a paid service, and the price depends on the number of web pages you plan to harvest.

- Zyte[6]: Formerly known as ScrapingHub, it is a web-based platform. Usually, enterprises use this tool to collect data from the web. For this, there will be a good amount of support from the developers' team. Furthermore, the company provides training for the enterprises that plan to use its harvesting tool. Zyte has a 14 days' free trial period.
- Octoparse[7]: It is a web-based tool for scraping data from the web. Generally, it is useful for collecting e-commerce data. The harvested data can be stored in different file formats such as Comma-Separated Values (CSV) and JSON. Octoparse has a free plan that allows you to build up to 10 crawlers. If you need more crawlers, there are other paid plans.

7.2 Harvesting Using Programming Languages

Most of the programmers or those who have some knowledge in programming prefer to use programming languages, such as Python and Java, to develop their web harvesters. There are many libraries available, and most of them are free to use. Each library offers a set of characteristics that specify how the library is harvesting the web and how it processes the harvested data. The main advantage of using programming languages for harvesting data from the web is that the harvested data has a much higher quality compared with the visual tools explained above. Below are some of the most common libraries that programmers use to develop web harvesters.

- Scrapy[8]: One of the most popular libraries used by programmers and developers to harvest data from web pages. It is written in Python. It is open-source, which means that it is free to use and modify. Scrapy is efficient when harvesting large amounts of data, and it is easy to understand and use.

[6] https://www.zyte.com/, last accessed: October 10, 2021.

[7] https://www.octoparse.com/, last accessed: October 10, 2021.

[8] https://scrapy.org/, last accessed: October 10, 2021.

- BeautifulSoup[9]: A library written in Python and easy to use. BeautifulSoup can parse only retrieved web pages. Therefore, you need to retrieve the web page first, then pass it to BeautifulSoup to start parsing it and extract information from the web page.
- Selenium[10]: It is a web harvesting tool written in Java. It also supports different programming languages such as Python and JavaScript. The main characteristic of Selenium is its capability of dealing with web pages that have dynamic content. It executes all the scripts before parsing them. This process will slow down the overall harvesting process, though, especially when harvesting a large number of web pages.
- Jsoup[11]: Jsoup is a Java library that is used to parse and extract information from HTML web pages. Jsoup is an open-source library. The library can retrieve the web page and then extract the elements. It is also efficient when a large amount of web pages needs to be harvested.

8 EduArc

The EduArc project[12] aims to provide a federated infrastructure for digital and open educational resources for teachers and students in Germany. We defined the requirements necessary to develop such an infrastructure from the teachers' point of view, which are the primary users of such infrastructure. Figure 4 shows the infrastructure of the EduArc project. The process starts with a set of crawlers that collect the metadata of educational resources from a set of repositories. Afterwards, a set of mappers will map the harvested data to the Common Data Model (CDM) of EduArc, which is designed based on the LOM standard. Then the mapped data is ready to be indexed in the Elasticsearch index. The search engine and the front end of EduArc allow the users to search the index and filter the results. Furthermore, the front end allows the users to add OER to the current index.

[9] https://www.crummy.com/software/BeautifulSoup/, last accessed: October 10, 2021.

[10] https://www.selenium.dev/, last accessed: October 10, 2021.

[11] https://jsoup.org/, last accessed: October 10, 2021.

[12] https://learninglab.uni-due.de/forschung/projekte/eduarc-digitale-bildungsarchitekturen, last accessed: October 10, 2021.

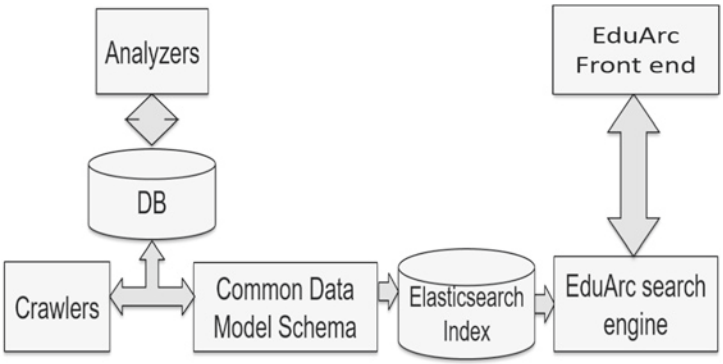

Fig. 4 The infrastructure of the EduArc project

9 Summary

In this chapter, we described the process of connecting the Open Educational Resources using the LOM standard. We illustrated the stages needed for such a process, starting from harvesting the data from the web pages to saving the results into a database or index. The harvesting process depends on the standard the OER providers used to model their metadata. The mappers also depend on the standard. The results can be stored in any format, such as JSON or CSV. We gave a brief description of the JSON format and its structure. Then, we listed some of the tools that help crawling the web pages. These tools can be visual, which does not require any knowledge in programming to run the harvesters. The other type of tool is used inside programming languages, which requires knowledge in programming. The latter tools render more high-quality data after harvesting the web pages compared to the visual tools. Finally, the EduArc project was described briefly to show the core concept and the workflow of the main infrastructure.

References

Hylén, J. (2006). Open educational resources: Opportunities and challenges. *Proceedings of open education*, vol. 4963, pp. 49–63, 01 2006.

IEEE (2020). IEEE standard for Learning Object Metadata. *IEEE Std 1484.12.1-2020*, pp. 1–50, 16 Nov. 2020.

Wilkinson, M. D., Dumontier, M., Aalbersberg, I. J., Appleton, G., Axton, M., Baak, A., ... & Mons, B. (2016). The FAIR guiding principles for scientific data management and stewardship. *Scientific data, 3*(1), 1–9.

Abdel-Qader, M., Saleh, A., Tochtermann, K. (2021). On the experience of federating open educational repositories using the learning object metadata standard. *EDULEARN21 Proceedings,* pp. 4819–4825.

Olston, C., & Najork, M. (2010). *Web crawling*. Now Publishers Inc.

Castillo, C. (2005). Effective web crawling. *Acm Sigir Forum, 39*(1), 55–56.

Johnson, J., Tsioutsiouliklis, K., & Giles, C. L. (2003). Evolving strategies for focused web crawling. In *Proceedings of the 20th international conference on machine learning (ICML-03)* (pp. 298–305).

Latif, A., Limani, F., & Tochtermann, K. (2021). On the complexities of federating research data infrastructures. *Data Intelligence, 3*(1), 79–87.

Nurseitov, N., Paulson, M., Reynolds, R., & Izurieta, C. (2009). Comparison of JSON and XML data interchange formats: A case study. *Caine, 9*, 157–162.

Divya, M. S., & Goyal, S. K. (2013). ElasticSearch: An advanced and quick search technique to handle voluminous data. *Compusoft, 2*(6), 171.