Building a Knowledge Graph of Licensed Educational Resources with Reification

Manoé Kieffer[1], Ginwa Fakih[1], Patricia Serrano-Alvarado[1], Margo Bernelin[2] and Colin De la Higuera[1]

[1]*Nantes University, LS2N, CNRS, UMR 6004, 44000 Nantes, France*

[2]*CNRS, Droit et Changement Social, UMR_C 6297, 44000 Nantes, France*

Abstract

This paper presents the ongoing construction of a knowledge graph (KG) of educational resources (ER) where RDF reification is essential. ERs can be described by their title, creator, license, etc., as well as the subjects they cover. Some subjects are the main focus of the ER, while others are only briefly discussed. To take into account this relevance, we propose to use RDF reification. Unfortunately, there is no established standard for reification, and various reification models exist, each with its own syntax and performance implications in terms of storage and query processing. The challenge we face is determining which reification model is best suited for our KG.

Keywords

Knowledge graph, educational resources, licenses, compatibility of licenses, RDF reification

1. Introduction

When teachers want to create a new course, they typically conduct a keyword search for (open) educational resources (ER) on the web to reuse and integrate into their course. While there are numerous valuable and relevant resources available (such as slides, videos, figures, text, code, etc.), many remain undiscovered because they are not well connected. Moreover, using these resources can present legal challenges if their licenses are not compatible with the course's license. To ensure the protection of all resources involved, the course's license should generally be less permissive. These legal issues can create barriers for both the teacher and the institution hosting the course. Ideally, the process of analyzing available resources to match a course plan and verifying licenses should not be time-consuming.

There are platforms to help learners to construct personalized learning paths[1], to enable teachers to share their ERs[2], and even to help teachers create new ERs[3]. However, a solution to help teachers to find relevant and license compatible ERs is missing. CC Search[4] can help users find images that are licensed under a Creative Commons license. Similarly, the Google

Fourth International Workshop On Knowledge Graph Construction, ESWC, 2023

Manoe.Kieffer@univ-nantes.fr (M. Kieffer); Ginwa.Fakih@univ-nantes.fr (G. Fakih); Patricia.Serrano-Alvarado@univ-nantes.fr (P. Serrano-Alvarado); Margo.Bernelin@univ-nantes.fr (M. Bernelin); cdlh@univ-nantes.fr (C. De la Higuera)

[1]https://labs.tib.eu/edoer/

[2]https://www.merlot.org/

[3]https://www.oercommons.org

[4]https://search.creativecommons.org/

search engine allows users to filter images by access rights based on licenses, including Creative Commons and other commercial licenses. However, these search engines do not provide suggestions for licenses that may protect an image mashup comprising multiple images protected by different licenses, nor do they return only images that are license-compatible. Furthermore, there are many open and free licenses[5], which are not considered in these solutions.

In the CLARA project, our goal is to design a solution that can identify a minimal, relevant set of educational resources with licenses that can protect the entire set of resources, whether or not the license is open. Our aim is to help teachers create content without having to focus on licensing aspects, and to avoid legal issues for their institutions.

ERs can be described by their title, creator, language, license, etc., as well as the subjects they cover. But not all subjects covered in an ERs are equally relevant. Some subjects are the main focus, while others are only mentioned briefly. Therefore, the relevance of each subject should be identified, and their relationship with educational resources should be weighed accordingly.

The best way to make ERs findable and reusable is to use the principles of the Linked Data. Semantic web technologies will allow a detailed description and interconnection of ERs thanks to well-known ontologies. Statement-level reification plays an important role as it allows annotating with scores or weights the relation of ERs and the subjects they treat. Most of the works using semantic web technologies propose tools to support learners with personalized learning [1], adaptive educational content [2], facilities for collaboration [3], etc., see [4, 5] for an overview of the state of the art. Few works assist teachers to find relevant educational resources, e.g., [6, 7], and to the best of our knowledge, there is no solution to help teachers create courses by reusing existing ones while preserving licenses and suggesting a license able to protect their courses.

The contributions of this paper are: (i) a methodology to build a knowledge graph (KG) of ERs using existing W3C recommendations (RDF, OWL, RML, SHACL, SPARQL) and well-known ontologies (DBpedia, Dublin Core, VoID, etc.); (ii) first experiments that evaluate and compare some representative reification models; (iii) first ideas about how to treat licence compatibility from an interdisciplinary point of view (Law and Computer Science).

The rest of this paper is organized as follows. Section 2 explains the methodology we follow to build the KG of ERs. Section 3 explains the pipeline used. Section 4 describes the experiments done to evaluate the reification models. Section 5 discusses legal aspects of our work concerning licenses. Finally, Section 6 outlines our future work and concludes.

2. Knowledge Graph description

Our project, named CLARA (Creating and Linking Licensable Educational Resources), aims to empower teachers to facilitate the creation of licensable ERs based on existing ones. The resources in our knowledge graph comprise unstructured ERs (documents, videos, and audio files, etc.), that are semantically annotated using the general-purpose ontology of DBpedia. By means of a wikification process, relevant DBpedia concepts related to ERs are used to provide a comprehensive description of each resource. This section introduces the CLARA ontology (Section 2.1), an explanation of the wikification process (Section 2.2), and statistics on our knowledge graph (Section 2.3).

[5]https://en.wikipedia.org/wiki/Free_license

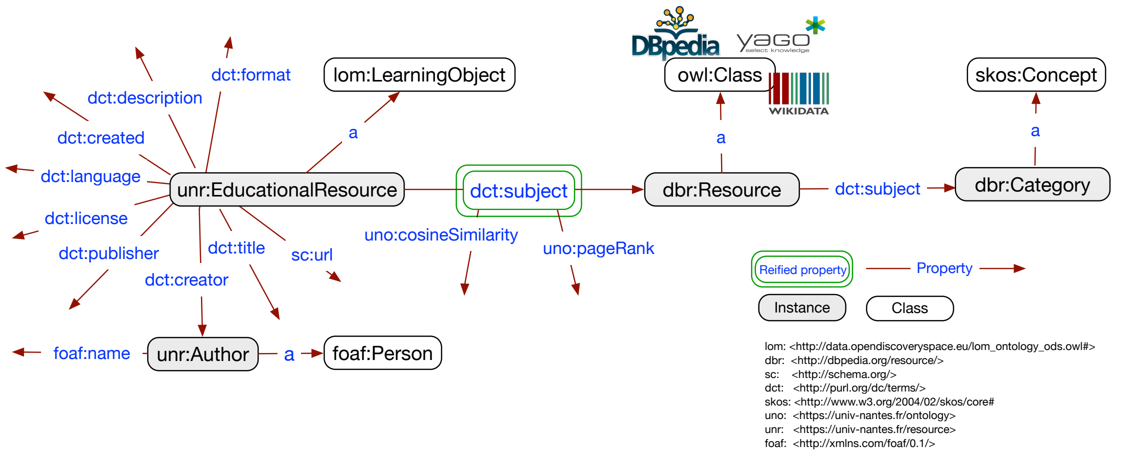


Figure 1: CLARA ontology

2.1. CLARA ontology

Figure 1 depicts the CLARA ontology diagram. Consistent with the ODS recommendation[6], we define ERs as LOM (Learning Object Metadata) Learning Objects. The LOM standard suggests a range of properties to describe learning objects, using common vocabularies such as Dublin Core and FOAF (dct:title, dct:creator, dct:language, dct:licence, dct:format, foaf:name, etc).

The particularity of our ontology lies in the extension of the LOM description to consider the subjects treated in the learning objects with relevance scores. To do this, we use RDF statement-level reification. Reification allows making statements about statements in a generic manner. In our case, it will allow to state that an ER treats a particular subject *to some extent*. Concretely, it will allow to annotate the predicate dct:subject in a fact (unr:EducationalResource, dct:subject, dbr:Resource) with relevance scores. In our project, these scores are determined using a wikification process, which identifies pagerank and cosine similarity values. More information on this process is provided in the next section.

Besides being reified, dct:subject is a multi-valued property, i.e., a subject-predicate pair having several objects. These objects are DBpedia resources which are instances of classes in DBpedia, Wikidata, and Yago ontologies. We consider also DBpedia categories, which are used in Wikipedia to organize articles and pages by subject matter. Since the goal of our KG is to identify ERs based on their subjects, DBpedia categories are essential. DBpedia resources are associated with their categories through the dct:subject property (dbr:Resource, dct:subject, dbr:Category).

2.2. Wikification of educational resources

Wikification is the process of automatic annotation of texts with Wikipedia entries. The goal is to bridge concepts identified in the text and Wikipedia articles. The wikification process generally involves two phases: term extraction and link disambiguation. There are various approaches to wikification that differ in the techniques used for extracting phrases and linking

[6]http://data.opendiscoveryspace.eu/ODS_LOM2LD/ODS_SecondDraft.html

them to external resources. CLARA uses a wikification tool called *Wikifier* which has shown a good performance compared to some state-of-the-art approaches [8][7]. This tool identifies *mentions* - phrases extracted from the input document - and uses them as hyperlinks between Wikipedia pages. The Wikipedia pages linked by a mention are considered as candidate concepts for that mention. Wikifier constructs a bipartite graph consisting of the mentions and their corresponding candidate concepts. The internal structure of hyperlinks between Wikipedia pages is then leveraged to weigh the edges (mentions) of the bipartite graph. A mention may have several candidate concepts because the same text can lead to different Wikipedia pages. To disambiguate mentions, the pagerank algorithm is applied over the graph. The concept with the highest pagerank score is selected for each mention, resulting in a set of Wikipedia concepts representing the input document. A threshold is then applied to retrieve the top-ranking concepts. In addition, Wikifier calculates the cosine similarity between the input document and the Wikipedia pages of the top-ranking concepts. To do this, the input document is represented as a bag-of-words model, where each term in the document is treated as a separate feature. Similarly, each Wikipedia page is represented as a bag-of-words model. The frequency of terms in the input document and each Wikipedia page is then compared to state each cosine similarity.

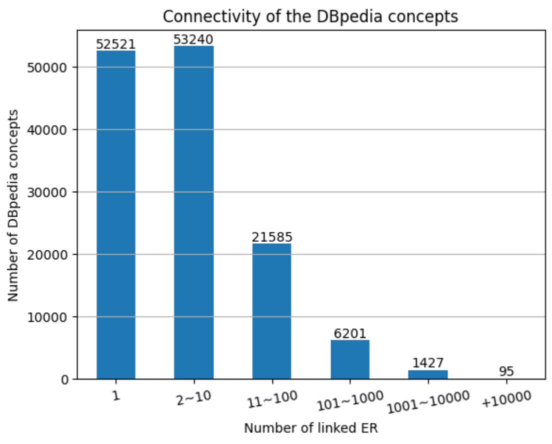
Currently, the CLARA project has collected a set of open educational resources that were wikified in the X5GON project[8]. X5GON was a Europe funded Horizon 2020 project whose goal was to build a cross modal, cross cultural, cross lingual, cross domain, and cross site global network of open educational resources. In a near future, we will exploit educational resources provided by the French Ministry of Education and Nantes Université.

2.3. Statistics and dataset content

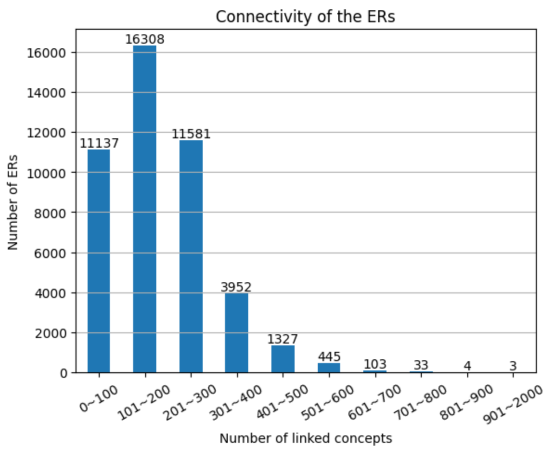
There exist several standard vocabularies to describe RDF datasets. The VoID vocabulary [9] is the most well-established vocabulary[9]. Additionally, the DCAT vocabulary [10] is a W3C recommendation to describe datasets, data services and data catalogs[10]. We provide a description of our KG using VoID and DCAT in a VoID file[11].The metadata for the dataset includes the label, license, SPARQL endpoint, provenance, prefixes used, and general statistics such as the number of triples, entities, subjects, objects, properties, etc. Our class partition currently consists of roughly 45K learning objects (lom:LearningObject)[12], 13K authors (foaf:Person), twelve licenses (odrl:Policy), and 2,193K categories (skos:Concepts)[13] along with 135K DBpedia resources that serve as the reified subjects of the CLARA ERs.

The distribution of concepts over ERs is far from being uniform. We consider the *connectivity of a concept* as its capacity to connect ERs. Figure 2a illustrates that over 100K concepts have poor connectivity. Roughly 52K concepts are associated with a single ER, while around 53K are connected to 2 to 10 ERs, as shown in the first two columns. The third column indicates that around 21K concepts connect between 11 and 1000 ERs. Lastly, the last column shows that 95

[7]https://wikifier.org
[8]https://www.x5gon.org
[9]https://www.w3.org/TR/void/
[10]https://www.w3.org/TR/vocab-dcat-2/
[11]https://gitlab.univ-nantes.fr/clara/pipeline/-/blob/main/statistics/clara-metadata.ttl
[12]From the X5GON dataset, we gathered only licensed resources.
[13]We obtained the entire hierarchy of DBpedia categories.



(a) Connectivity of the DBpedia concepts



(b) Distribution of DBpedia concepts.

Figure 2: Statistics of the DBpedia Concepts

concepts have very high connectivity, being used in more than 10K ERs. The distribution of concepts by ER is shown in Figure 2b. The first three columns show that the majority of ERs are associated with less than 300 concepts. On the other hand, the last three columns demonstrate that only a small number of ERs (40 ERs) are linked to a considerable number of concepts.

The pagerank score of a concept is local to an ER, it depends on the number of concepts that the Wikifier associates with this ER. The sum of the pagerank values of all concepts linked to an ER is 1. Thus, the greater the number of concepts, the lower their pagerank score. To determine the global relevance of a concept over the KG, we use the number of associated ERs and their pagerank scores. We calculate the relevance of a concept as the ratio of all the pagerank scores of that concept over all the pagerank scores in the KG, i.e., $relevance(c) = \frac{\sum_{\mathrm{i}} \mathrm{PR}(ER_{\mathrm{i,c}})}{\sum_{\mathrm{y}} \sum_{\mathrm{i}} \mathrm{PR}\left(\mathrm{ER}_{\mathrm{i,C_y}}\right)}$.

3. Data transformation pipeline

Figure 3 shows the pipeline for the CLARA ETL process. The extraction phase involves collecting data from a Postgres database. In the transformation phase, JSON files are converted into semantic RDF triples. To compare different RDF reification models, we created four RML mappings to obtain standard reification, RML-star, named graphs, and singleton properties. We used SHACL shapes to validate our RDF graphs. In the loading phase, we loaded and set up two SPARQL endpoints: Virtuoso and Jena. The rest of this section focus on the RML mappings that are available on GitLab[14].

3.1. Reification models

Before describing our RML mappings, in particular the RDF reification, we briefly overview the four reification models that we implement and evaluate.

[14]https://gitlab.univ-nantes.fr/clara/pipeline

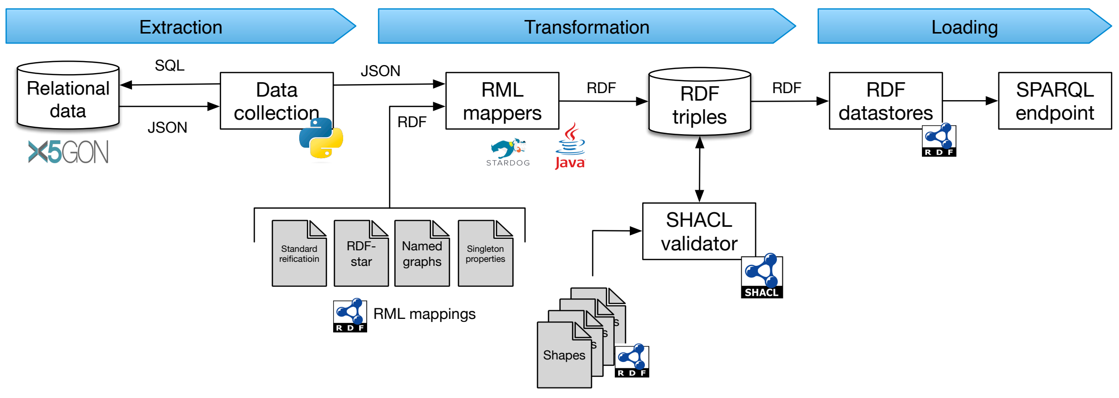


Figure 3: Pipeline of the CLARA ETL (Extract, Transform, Load) process.

Standard reification. The standard reification model was proposed within RDF primer standardised by W3C [11][15]. In this model, rdf:Statement is used to define the triple that will be annotated (rdf:subject, rdf:predicate, and rdf:object). The defined statement can be identified by a blank node or a URI. Listing 3.1 gives the representation in RDF triples As we can see, our annotations are two score values that are linked to the statement.

```
uns:S100   rdf:type  rdf:Statement  ;
           rdf:subject  unr:ER  ;
           rdf:predicate  dc:subject  ;
           rdf:object  :Query_Language  ;
           uno:cosineSimilarity  ”0.6”  ;
           uno:pageRank  ”0.4”  .
```

Listing 1: Standard reification in Turtle.

Named graphs. Carroll et al. [12] proposed an extension to the RDF data model that allows RDF graphs to be named by URIs, which are referred to as named graphs. In this approach, a named graph is represented as a pair (g, n), where g is an RDF graph and n is an IRI, a blank node, or a default graph. The statements to be annotated are defined in one RDF graph, while the annotations themselves are defined in another RDF graph. The annotations are directly linked to this graph. Listing 2 shows the syntax for this example.

```
unr:ER  dc:subject  :Query_Language  <g-100>  .
<g-100>  uno:cosineSimilarity  ”0.6”  ;
         uno:pageRank  ”0.4”  .
```

Listing 2: Named Graphs in N-quads.

Singleton properties. The singleton property model [13] proposes creating a unique property for every triple that has associated metadata. In this model, a new node is created to represent the new property, which is connected directly to the original annotation property using the proposed property *singletonPropertyOf*. The same property is used for all metadata associated with a statement. An example of RDF triples using this model is shown in Listing 3.

```
unr:ER    <p-200>  :Query_Language  .
```

[15]https://www.w3.org/TR/rdf-primer/#reification

```
<p-200>  rdf:singletonPropertyOf dc:subject;
         uno:cosineSimilarity "0.6" ;
         uno:pageRank "0.4" .
```

Listing 3: Singleton properties in Turtle.

RDF-Star. Recently, [14][16] proposed RDF-star and SPARQL-star as extensions to RDF and SPARQL to enable graph nesting and simplify the representation of reified statements. RDF-star and SPARQL-star allow for the recursive nesting of graphs, eliminating the need for declaring edge identifiers that are linked with metadata. RDF-star enables the nesting of triples within other triples as subjects or objects by using double angle brackets ≪ ≫. As a result, every reified statement can be interpreted as a single RDF triple. An example of RDF-star reification is shown in Listing 4.

```
<< unr:er dc:subject :Query_Language >> uno:cosineSimilarity "0.6" ;
                                        uno:pageRank "0.4" .
```

Listing 4: RDF-Star in Turtle.

3.2. RML mappings

We define our four mappings using the RML language [15]. In Listing **??** we show an excerpt of the RML mapping used to transform the JSON data in triples with standard reification. This excerpt contains the RML rule that generates the annotations. In Lines 2 to 6, the rule iterates over every concept of every ER. In Lines 7 to 9, the subjects are defined as IRIs of type rdf:Statement, following the shape used in standard reification. The following lines define the predicates and objects related to these subjects. Lines 25 to 29 define the annotation of the pagerank score, taken from the JSON under the attribute "norm_pageRank".

The RML mapping for the singleton property transformation is very similar to the standard reification one. It defines subjects as IRIs related to dct:subject via the rdf:singletonProperty property. Subjects are then defined as properties for the reified triples. Listing 6 shows the excerpt of our mapping defining this part of the rule.

The key difference in the RML mapping for named graphs is the creation of a quad for each fact to be annotated. Our mapping uses the property rr:graphMap to specify the IRI of the graph containing the triple to be reified. Listing 7 shows how we define the predicate, object, and graph for each ER as a subject.

For the RDF-star approach, the mapping is more complex and requires the use of RML-star [16], an extension of RML. We first define the triple that needs to be reified and then use it as the subject in the rule that defines the annotation. Listing 8 shows an excerpt of our mapping, where the property rml:quotedTriplesMap maps the subject of the annotations to another RML rule (the one defining the triple to reify).

To generate RDF triples we use two mappers. For standard reification, named graph, and singleton property, we use an RML mapper implemented in Java[17]. To generate RDF-star triples, we used morph-KGC [17], which can function as both an RML mapper and an RML-star mapper.

```
1  <ReifiedMapping> a rr:TriplesMap;
2      rml:logicalSource [
```

[16]https://w3c.github.io/rdf-star/cg-spec/2021-12-17.html
[17]https://github.com/RMLio/rmlmapper-java

```
        rml:source "json/ER/ER_0.json";
        rml:referenceFormulation ql:JSONPath;
            rml:iterator "$.resources[*].concepts[?(@.dbPediaIri)]";
        ];
        rr:subjectMap [
            rr:template "https://clara.univ-nantes.fr/statement/{er_id}-{c_id}";
            rr:class rdf:Statement;
        ];
        rr:predicateObjectMap [
            rr:predicate rdf:subject;
            rr:objectMap[rr:template "https://clara.univ-nantes.fr/resource/{er_id}";
                         rr:termType rr:IRI;   ];
        ];
        rr:predicateObjectMap [
            rr:predicate rdf:predicate;
            rr:objectMap [ rr:constant dct:subject ];
        ];
        rr:predicateObjectMap [
            rr:predicate rdf:object;
            rr:objectMap [  rml:reference "dbPediaIri";
                            rr:termType rr:IRI;     ]
        ];
        rr:predicateObjectMap [
            rr:predicate uno:pageRank;
            rr:objectMap [  rml:reference "norm_pageRank";
                            rr:datatype xsd:double;     ]
        ].
```

Listing 5: Excerpt of the RML mapping for standard reificaiton.

```
rr:predicateObjectMap [
    rr:predicateMap [
        rr:template "https://clara.univ-nantes.fr/statement/{er_id}-{c_id}";
    ];
    rr:objectMap [  rml:reference "dbPediaIri";
                    rr:termType rr:IRI;     ]
].
```

Listing 6: Excerpt of the RML mapping for singleton property.

```
rr:predicateObjectMap [
    rr:predicate dct:subject;
    rr:objectMap [  rml:reference "dbPediaIri";
                    rr:termType rr:IRI;     ];
    rr:graphMap [
        rr:template "https://clara.univ-nantes.fr/statement/{er_id}-{c_id}";      ];
];
```

Listing 7: Excerpt of the RML mapping for named graph.

```
rml:subjectMap [
    rml:quotedTriplesMap :ER_concept_link;
];
```

Listing 8: Excerpt of the RML-star mapping for RDF-star.

4. Experimental evaluation of reification models

The goal of this section is to compare the different reification models using Virtuoso and Jena. Section 4.1 describes the differences of the different approaches in terms of storage space and ease of use. And Section 4.2 shows the experiments carried out to evaluate the performance of the different approaches in terms of query performance.

4.1. Syntax comparison of reification models to define annotated triples

The described reification models differ in various criteria such as the number of triples, human understandability, flexibility, and syntax support.

Number of triples. Standard reification is the most costly approach since it needs five triples for each reified statement. Singleton properties models needs three triples. Named graphs and RDF-star are the most compact models requiring two triples. One advantage of named graphs is that reification can be defined not only at statement level but also for a group of triples or even a dataset. This model is the only one that allows for different levels of granularity for annotations.

Human understandability. Standard reification and singleton properties can be difficult for humans to understand because the reified statement is split into multiple forms, making it challenging to comprehend the statement as a whole. Named graphs provide a clearer representation since they maintain the direct links between the terms of the reified statement. RDF-star was proposed as a solution to simplify statement-level annotations by providing a clear representation of statements and their annotations without the need for refactoring the statement, declaring new predicates, or new classes. This ensures easier human understanding of the data.

Flexibility. All of these reification models are flexible when it comes to adding new annotations to an already reified statement. Adding new annotations only requires adding one additional triple for each approach. Additionally, all of these models support multi-valued properties, which are annotations that have a subject-predicate pair with several objects.

Syntax support. Standard reification and singleton properties conform to the core RDF model proposed in 2004. Named graphs represent an extension to the triple RDF model and is part of the standard RDF1.1, which was published in 2014. RDF-star proposes to extend the RDF specification further. While all of these models are supported in the SPARQL standard, RDF-star proposes SPARQL-star as a query language. However, several RDF stores, such as Jena, Stardog, RDF4J, BlazeGraph, AllegroGraph, etc., now support the implementation of RDF-star and SPARQL-star[18].

We made a first experimental evaluation of the four reification models to compare the number of triples and the database size to store our KG using two triplestores, Virtuoso and Jena. As we show in Table 1, named graphs and RDF-star models are the most compact in terms of the number of triples required for our dataset, with only 37M triples. Standard reification and singleton properties require a higher number of triples, with 70M and 53M respectively. We consider the storage of the triples without reification, i.e., (unr:ER, dct:subject, dbr:Concept) for all models except for named graphs. For instance, in RDF-star, the triple that is used as

[18]https://w3c.github.io/rdf-star/implementations.html

	Standard reification		Singleton Properties		Named graphs		RDF-Star	
	#Triples 10^6	DB size 10^9	#Triples 10^6	DB size 10^9	#Triples 10^6	DB size 10^9	#Triples 10^6	DB size 10^9
Virtuoso	70,1	3.1	53,5	3.1	37,0	3.0		
Jena	70,1	56	53,5	51	37,0	50	37,0	49

Table 1
Generated number of triples and DB size of different reification models for the CLARA KG.

subject is stored in one triple. Concerning the database size, Jena with standard reificaiton is the largest with 56GB. In general, databases under Jena are larger than Virtuoso. Virtuoso has little differences in the storage size.

4.2. Experimental comparison of query performance

Experiments were run on a virtual machine with 128Go of RAM, 2GHz with 32 cores, on a Debian GNU/Linux 11 (bullseye). All tests were run using docker images of the query engines[19] [20]. All engines were given access to 16Go of RAM, and the test were run in isolation, with caches on.

Sets of queries. Based on series of queries A, B and F used in [18], we define 4 query templates that we present as SPARQL-star queries for simplicity and clarity. Query 1 gets the list of concepts and all the associated hierarchy of categories for a given ER (Listing 9). Given an ER, query 2 gets the associated concepts with a pagerank score greater than a given threshold (Listing 10). Query 3 gets ERs associated to a set of 3 given concepts (Listing 11). Query 4 is a typical query used to retrieve data in the CLARA interface, given a set of DBpedia concepts, it gets the ERs and their descriptions (Listing 12).

Query templates are grounded with instances selected beforehand. We chose two ERs with little connectivity (2 to 10 connections), two with medium connectivity (100 to 300 connections), and two with large connectivity (more than 1000 connections). For DBpedia concepts, we chose two concepts with a connectivity in the range 2 to 10, two in the range 100 to 2000, and two with over 10000 connections.

Procedure. Queries were sent to the SPARQL endpoints of the two query engines using a python script and the library SPARQLWrapper. Table 2 shows the results of our experiments. Each query was executed 5 times and the average execution time is displayed. The first row shows the average execution time of executing queries with all the chosen instances of each query template. The second row shows the average execution time of of queries grounded the instance (ER or concept) with the lowest connectivity. The Third column shows the average execution time of queries grounded to the instance (ER or concept) with the largest connectivity. Results for Query 4 are calculated differently. First row corresponds to the average execution time of the grounded to the six instances of concepts. The two other rows are similar to the other queries.

Analysis of results. The execution of Query 1 finished prematurely due to the amount of memory the property path query takes when navigating the DBpedia categories. For both

[19]https://hub.docker.com/r/secoresearch/fuseki
[20]https://hub.docker.com/r/tenforce/virtuoso

query engines, only the three smallest grounded instances succeeded. In the case of Virtuoso, the query had to be adapted using the option t_distinct to improve its efficiency with star property paths. Standard reification and named graph approaches exhibit similar behavior in Jena and Virtuoso, except for Query 1 where Virtuoso is superior on average. Singleton properties performs especially badly with Jena, but with Virtuoso it performs similarly to the other options. RDF-star with Jena proves to be the slowest approaches for our graph. It is impossible to know if this is due to how Jena handles RDF-star or if this issue would be present with other engines, further tests are needed. Overall, when comparing the performance of all reification models in terms of execution time, it appears that standard reification and named graphs are the best, followed by singleton properties, and finally RDF-Star. And Virtuoso seems to be performing equally well or better than Jena no matter the reification approach.

```
SELECT *
WHERE {
    [ER] dct:subject ?concept .
    ?concept dct:subject/skos:broader*
        ?categorie .
}
```

Listing 9: Query template 1.

```
SELECT *
WHERE {
    << ?er dct:subject [concept] >> uno
        :pageRank ?pr .
    FILTER(0.01 < ?pr) }
```

Listing 10: Query template 2.

```
SELECT *
WHERE {
    <<?er dct:subject [concept]>> uno:cosineSimilarity ?cosine1 ;
                                  uno:pageRank ?pr1 .
    <<?er dct:subject [concept]>> uno:cosineSimilarity ?cosine2 ;
                                  uno:pageRank ?pr2 .
    <<?er dct:subject [concept]>> uno:cosineSimilarity ?cosine3 ;
                                  uno:pageRank ?pr3 . }
```

Listing 11: Query template 3.

```
SELECT *
WHERE {
    << ?er dct:subject ?concept >> uno:pageRank ?pr .
    ?er dct:language ?language ;
        dct:license ?license ;
        dct:format ?format ;
        dct:title ?title ;
        schema:url ?url .
    OPTIONAL {  ?er dct:creator ?author .           }
    OPTIONAL {  ?er dct:description ?description .  }
    VALUES ?concept {   [set_of_dbpedia_concepts]   }
    VALUES ?format  { "application/pdf" "video/mp4" }   }
```

Listing 12: Query template 4.

5. Compatibility of licenses when reusing licensed educational resources

Under law, a license is a unilateral contract allocating rights and duties over an intellectual creation. In practice, the terms of the license will outline the permissions, prohibitions, and

Query engine	Standard reification				Singleton properties			
	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4
	10.12	0.06	0.65	2.82*	11.60	1.27	1.21	2.88*
Virtuoso	4.73	0.001	0.03	0.01	7.27	2.40	1.05	0.15
	OoM	0.047	1.63	1.57	OoM	0.21	1.60	1.77
	28.42	0.04	0.13	3.30*	33.94	26.65	43.85	374.88*
Jena	3.01	0.01	0.01	0.01	3.62	26.26	42.35	60.21
	OoM	0.12	0.36	2.11	OoM	27.33	46.84	71.18

Query engine	Named graphs				RDF-Star			
	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4
	10.84	1.08	0.54	2.77*				
Virtuoso	6.68	2.40	0.46	0.15		Not	supported	
	OoM	0.68	1.09	2.30				
	33.33	0.24	0.04	3.59*	92.45	134.02	139.66	7.57*
Jena	3.74	0.019	0.026	0.02	3.99	142.52	140.69	14.08
	OoM	0.74	0.08	2.88	OoM	126.33	144.42	4.82

Table 2
Query execution time for the four reification models (OoM:Out of Memory).

obligations associated with the reuse of the intellectual work covered. The license is chosen by the author of the work and will be binding for anyone who wishes to reuse the said work.

Machine-readable licences. RDF metadata allow to define the content's license, for example RDFa[21] allow to specify the license of a web page using RDF embedded in HTML. This information helps search engines to know which license protects a resource which can then help them better recommend resources. Machine-readable licenses are licenses translated to a language, that allows the representation of licenses and their terms. For example, CC represents their licenses in RDF using ccREL [19]. Our project builds upon the use of machine-readable licenses by allowing research through licensed resources according to the semantically represented terms of their license.

Compatibility and compliance. The notion of license compatibility is important when planning to reuse resources. To integrate or include a resource in a new one, a teacher must verify if the licenses of the resources used are compatible with the license of the future resource. As CLARA aims to ease the search and reusability of resources, we need to integrate this notion to further help teacher with the creation of new resources. This project bases itself on the project CaLi [20]. CaLi compares licenses which have the same actions and that either allow, forbid, or oblige those actions. Depending on whether two licenses allow, forbid, or oblige its actions, and depending on a lattice of the restrictiveness of those possible cases, CaLi can compare licenses in terms of their restrictiveness. As a result CaLi can partially orders a set of licenses using a restrictiveness relation. Then CaLi refines that partial order of restrictiveness of licenses using constraints in order to identify which licenses are compatible with each other.

CaLi defines compliance and compatibility this way. *A license l_j is compliant with a license l_i if a resource licensed under l_i can be licensed under l_j without violating l_i. If a license l_j is compliant with l_i then we consider that l_i is compatible with l_j and that resources licensed under l_i are reusable with resources licensed under l_j.* This means that CaLi does not define compatibility (or compliance) as a symmetrical relation. In addition, usually but not always, when l_i is less restrictive than l_j

[21]https://www.w3.org/TR/rdfa-primer/

then l_i is compatible with l_j. And in every case, when l_i is less restrictive than l_j then l_j cannot be compatible with l_i.

5.1. Source of license incompatibilities

License incompatibility is a complex issue as it can arise from multiple places [21, 22]. Firstly, a license can be incompatible with another license because their clauses contradict each other. And a license could also be incompatible with another version of itself for multiple reasons, we will discuss those reasons in the following sections.

Between different licenses. The first and most obvious reason a license can be incompatible with another is due to contradictions in their different terms, i.e., duties, permissions or prohibitions. For example, the Creative Commons license CC BY-NC is incompatible with the Creative Commons license CC BY, as CC BY-NC prohibits the commercial use of the licensed content when CC BY allows it. Note that this does not imply CC BY is incompatible with CC BY.

Between different versions of the same license. Incompatibility may also arise for different versions of a same license. An updated version of a license could be more or less restrictive than its counterpart. Take the case of MPL-1.1 and MPL-2.0 like discussed in [23]. [23] considers MPL-1.1 and MPL-2.0 as non-transitivly compatible. Using the definition of CaLi, MPL-1.1 would be considered as incompatible with MPL-2.0 because MPL-2.0 is less restrictive.

Moreover, the translation of a license will also result in the creation of a different version of the same license, this time with linguistic variations. Translating licenses to other languages is an important topic, especially when licenses aim at being open and international. But translating a license to another language can cause imprecision and small inconsistencies. That consequence is inherent to the work of translating text and even small changes in phrasing or words could cause different interpretations to arise [21, 22]. Which in turn could make two licenses incompatible as they would not contain the same terms anymore.

Another source of incompatibility discussed here is the incompatibility born from a license adapted to be applied in multiple legal orders, that is to say adapted to fit multiple national Laws. This task comes with legal variation as well as linguistic translation. It can lead to differences in the terms of the license' versions, and to versions of a license that are not concerned with the same national law anymore, possibly making them incompatible.

The last possible incompatibility with licenses versions discussed here are machine-readable versions. This kind of version is related to translation but it is a translation to informatic models and not another human language. The same way differences in words and phrasing can cause incompatibility between two translations of a license, differences between a license and its machine readable version can also create inconsistencies. As our project relies on the machine-readability of licenses directly, this case is not taken in the scope of the project.

5.2. Handling compatibility in CLARA

In order to better fulfill its goal of helping teachers to find educational resources, the CLARA project should take into consideration the concepts of compatibility and incompatibility of licenses. This will allow CLARA to recommend more resources fitted to the teacher's needs, and in particular, it should be able to query resources that are compatible with the license that the teacher will use for their own resource. In the case where the teacher does not specify a license for their future course but still selects a list of resources to build their own new resource. The

CLARA interface needs to be able to achieve two things. First, it should be able to verify that all the licenses of the selected resources can be compatible with a single one. Then it should be able to suggest a set of licenses that fulfill that criterion so that the teacher can select one for their own resource. Secondly, if all the licenses of the selected resources cannot be compatible with a single resource, the interface should propose a way to resolve that conflict by pointing out resources to remove from the set.

In order for the CLARA project to achieve those goals, the KG and the interface need to be built by taking into consideration both the concept of compatibility as defined by CaLi and the challenges discussed in Section 5.1. To guarantee the biggest impact, it is important to select as many licenses that share the same actions (in order to be comparable in terms of restrictiveness and to be comparable using the logic of CaLi). But it is also important to minimize the chance that those licenses end up being incompatible for the reasons explained in Section 5.1.

6. Conclusion and future work

In this paper, we report our current work on setting up a knowledge graph of educational resources with a significant number of reified triples. We focused on the used pipeline and showed some preliminary experiments.

Our current work involves evaluating four reification models in the RDF4J triplestore. We also plan to assess other reification models, such as n-ary relations. Additionally, we are planning to perform entity resolution for authors, using external knowledge sources like Google Scholar and ORCID for author name disambiguation. Moreover, we are planning to expand our knowledge graph by adding more datasets to it. Our goal is to collect ERs from the French Ministry of Education, as well as from Nantes University.

Since our experimental evaluation is still ongoing, the current CLARA SPARQL endpoint uses standard reification. We have developed a preliminary version of a web application that utilizes the CLARA SPARQL endpoint as its back-end[22]. The current version of the web app functions as a keyword-search engine, allowing users to retrieve educational resources based on the concepts they cover. In the near future, we plan to add graph exploration capabilities, which will enable users to navigate among ERs and their concepts. Additionally, we aim to include a feature that will allow teachers to select a set of ERs for reuse in a new course and suggest licenses that can protect the new course.

In our upcoming scientific work, we plan to focus on SPARQL query relaxation in the presence of reification. When searching for educational resources on specific subjects, a query may yield empty or insufficient results. One effective solution to this problem is to dynamically expand the scope of the query by relaxing the constraints in the query. The goal is to generalize the user query to produce alternative results that are similar to those expected by the initial query. However, applying query relaxation to queries that involve both data and metadata constraints poses several challenges [24]. Current relaxation techniques are not designed for querying metadata, as they do not take metadata values into account in the query ranking function that gathers the k-relevant answers closest to the original query. Additionally, current relaxation techniques are defined for equality of values, whereas metadata constraints may include intervals of values in a range. Syntaxes used to represent RDF reification and to query

[22]https://clara.univ-nantes.fr/

reified triples may also result in unexpected behavior when using current query relaxation approaches. Therefore, we believe that new query relaxation techniques should be developed to deal with queries that involve both data and metadata.

Acknowledgments

This work has received a French government support granted to the Labex Cominlabs excellence laboratory and managed by the National Research Agency in the "Investing for the Future" program under reference ANR-10-LABX-07-01. Authors thank Master students in Computer Science of Nantes University for her participation in some aspects of this work.

References

[1] N. Henze, P. Dolog, W. Nejdl, Reasoning and ontologies for personalized e-learning in the semantic web, Journal of Educational Technology & Society 7 (2004).

[2] R. Denaux, V. Dimitrova, L. Aroyo, Integrating Open User Modeling and Learning Content Management for the Semantic Web, in: International Conference on User Modeling, volume 3538, 2005.

[3] K. Halimi, H. Seridi-Bouchelaghem, Semantic Web Based Learning Styles Identification for Social Learning Environments personalization, in: Web Intelligence, volume 13, 2015.

[4] J. Jensen, A Systematic Literature Review of the Use of Semantic Web Technologies in Formal Education, British Journal of Educational Technology 50 (2019).

[5] G. Vega-Gorgojo, J. I. Asensio-Pérez, E. Gómez-Sánchez, M. L. Bote-Lorenzo, J. A. Muñoz-Cristóbal, A. Ruiz-Calleja, A Review of Linked Data Proposals in the Learning Domain, J. Univers. Comput. Sci. 21 (2015).

[6] J. Atkinson, A. Gonzalez, M. Munoz, H. Astudillo, Web Metadata Extraction and Semantic Indexing for Learning Objects Extraction, Applied Intelligence 41 (2014).

[7] N. Shabir, C. Clarke, Using Linked Data as a Basis for a Learning Resource Recommendation System, in: Workshop on Semantic Web Applications for Learning and Teaching Support in Higher Education, 2009.

[8] J. Brank, G. Leban, M. Grobelnik, Semantic annotation of documents based on wikipedia concepts, Informatica 42 (2018).

[9] K. Alexander, R. Cyganiak, M. Hausenblas, J. Zhao, Describing linked datasets., in: Workshop on Linked Data on the Web (LDOW), volume 538, 2009.

[10] R. Albertoni, D. Browning, S. Cox, A. Gonzalez-Beltran, A. Perego, P. Winstanley, et al., Data catalog vocabulary (dcat)-version 2, World Wide Web Consortium (2020).

[11] F. Manola, E. Miller, B. McBride, et al., RDF primer, W3C recommendation 10 (2004).

[12] J. J. Carroll, C. Bizer, P. Hayes, P. Stickler, Named graphs, Journal of Web Semantics (JWS) 3 (2005).

[13] V. Nguyen, O. Bodenreider, A. Sheth, Don't like RDF reification? Making statements about statements using singleton property, in: Proceedings of the 23rd international conference on World Wide Web (WWW), 2014.

[14] O. Hartig, Foundations of RDF* and SPARQL*:(An alternative approach to statement-level metadata in RDF), in: International Workshop on Foundations of Data Management and the Web (AMW), volume 1912, 2017.

[15] A. Dimou, M. Vander Sande, P. Colpaert, R. Verborgh, E. Mannens, R. Van de Walle, RML: A generic language for integrated rdf mappings of heterogeneous data., Workshop on Linked Data on the Web (LDOW) 1184 (2014).

[16] J. Aenas-Guerrero, A. Iglesias-Molina, O. Corcho, D. Chaves-Fraga, T. Delva, A. Dimou, Rml-star: A declarative mapping language for rdf-star generation, 2021.

[17] J. Arenas-Guerrero, A. Iglesias-Molina, D. Chaves-Fraga, D. Garijo, O. Corcho, A. Dimou, Morph-kgc star: Declarative generation of rdf-star graphs from heterogeneous data, Submitted to Semantic Web Journal (SWJ) (2022).

[18] F. Orlandi, D. Graux, D. O'Sullivan, Benchmarking rdf metadata representations: Reification, singleton property and rdf, in: International Conference on Semantic Computing (ICSC), 2021.

[19] H. Abelson, B. Adida, M. Linksvayer, N. Yergler, 10. cc rel: The creative commons rights expression language (2008).

[20] B. Moreau, P. Serrano-Alvarado, M. Perrin, E. Desmontils, Modelling the compatibility of licenses, in: Extended Semantic Web Conference (ESWC), 2019.

[21] M. Bernelin, The compatibility of open/free licences: a legal imbroglio, International Journal of Law and Information Technology 28 (2020).

[22] M. D. de Rosnay, Creative commons licenses legal pitfalls: Incompatibilities and solutions, Technical Report, University of Amsterdam, Institute for Information Law, 2009.

[23] G. M. Kapitsaki, F. Kramer, N. D. Tselikas, Automating the license compatibility process in open source software with spdx, Journal of systems and software 131 (2017).

[24] G. Fakih, P. Serrano-Alvarado, A survey on sparql query relaxation under the lens of rdf reification, Submitted paper to the Semantic Web Journal (2023).