Metadata as Linked Open Data: mapping disparate XML metadata registries into one RDF/OWL registry

Marta Villegas, Maite Melero, Núria Bel
Universitat Pompeu Fabra
Roc Boronat, 138 - 08018 Barcelona
E-mail: {marta.villegas, maite.melero, nuria.bel}@upf.edu

Abstract

The proliferation of different metadata schemas and models pose serious problems of interoperability. Maintaining isolated repositories with overlapping data is costly in terms of time and effort. In this paper, we describe how we have achieved a Linked Open Data version of metadata descriptions coming from heterogeneous sources, originally encoded in XML. The resulting model is much simpler than the original XSD schema and avoids problems typical of XML syntax, such as semantic ambiguity and order constraint. Moreover, the open world assumption of RDF/OWL allows to naturally integrate objects from different schemas and to add further extensions, facilitating merging of different models as well as linking to external data. Apart from the advantages in terms of interoperability and maintainability, the merged repository enables end-users to query multiple sources using a unified schema and is able to present them with implicit knowledge derived from the linked data. The approach we present here is easily scalable to any number of sources and schemas.

Keywords: LOD, RDF/OWL, metadata, META-SHARE

1. Motivation

Because of our participation in different projects (CLARIN, Panacea, Metanet4U) we have different registers and catalogues. They all resemble in that they provide curated metadata descriptions for a variety of NLP resources (including lexica, corpora and NLP services & workflows) and give access to them. They essentially differ in the metadata model used. This results in a frustrating lack of interoperability among them. Broadly speaking, the whole picture goes as follows:

- Web Services (WS) are described and registered in the IULA WS registry[1] which is an instance of the BioCatalogue application[2]. The BioCatalogue is a registry that allows to: register, discover, annotate, monitor and execute WS. It is integrated with Taverna which facilitates integration into NLP pipelines. Records are encoded following the BioCatalogue schema.
- Workflows (that is, pipelines of NLP WS) are registered and described in myExperiment catalogue[3]. MyExperiment makes it easy to find, use and share scientific workflows and to build communities. It is also integrated with Taverna [4] and linked to BioCatalogue. Records in the catalogue are encoded following the MyExperiment model.
- As META-SHARE (MS) members, we set up and maintain our own Language Resource Repository Node[5] which is synchronized with the MS network. All metadata records in the repository are described following the MS schema. Our MS node contains lexica and corpora.
- Finally, we have an OAI-PMH server[6] that includes all records from the catalogues above (that is: service, workflow and resource descriptions). The goal of the OAI-PMH server is to collect and expose all metadata from our specialised catalogues following the standard harvesting protocol. Our server is harvested by the Virtual Language Laboratory[7], by the UPF Digital Repository[8] and by OLAC[9] (Open Language Archives Community) among others.
- The institutional repository of the UPF [10] collects, disseminates and preserves in digital form the intellectual output that results from the academic and research activity of the University. This includes not only traditional publications but also primary data. Thus, most of the primary data described in our MS records are stored in this repository and many publications (articles, documentation, manuals, etc) are also there. The e-Repository builds on DSpace and uses Dublin Core (DC) metadata. The e-Repository offers us long term preservation facilities and persistent identifiers (handlers).

The scenario described above illustrates well the diversity of formats that metadata for language resources may take. The proliferation of different metadata schemas and models pose serious problems of interoperability (Haslhofer and Klas, 2010). In addition, there is no connection between entities in the different catalogues, although in many cases they may just be one and the same. Maintaining isolated repositories with overlapping data is a waste of time and effort, as we need to curate the same data in different places and in different formats.

2. Objectives

In this paper we will describe how we have achieved a Linked Open Data (LOD) version of metadata

[1] http://services.iula.upf.edu/
[2] https://www.biocatalogue.org/
[3] http://myexperiment.elda.org/
[4] http://www.taverna.org.uk/
[5] http://metashare.upf.edu/
[6] http://ws02.iula.upf.edu/corpus_data/oai-iula/?verb=Identify
[7] http://catalog.clarin.eu/vlo/?fq=collection:IULA-UPF+Centre+de+Compet%C3%A8ncia+CLARIN
[8] http://repositori.upf.edu/
[9] http://www.language-archives.org/archive/iula.upf.edu
[10] http://repositori.upf.edu/

descriptions coming from heterogeneous sources. We will see that the LOD approach not only proves essential for linking data but also improves modeling and promotes interoperability between the different models (Lam et al., 2006). Thus, we were able to match, merge and link three different metadata schemas, providing a unique repository which allows users querying multiple sources using a unified schema rather than querying each source separately. The ultimate goal of this task was to produce a registry of NLP assets that helps researchers in Humanities and Social Sciences to find relevant resources and services for their research. Since our users are unaware of the possibilities that NLP can offer them, we could not expect them to use the catalogue as a way to find what they already knew there existed, but rather as a place to discover things they did not know beforehand. The approach presented here is easily scalable to any number of sources and schemas.
The reminder of this paper is organized as follows: In section 3 we describe the problems faced and decisions taken when RDFying XML data. We take the MetaShare (MS) schema as a convenient example of an XML based model. We choose the MS model because it is a large and exhaustive schema to be used eventually by a wide community. In section 4 we describe the process of linking the database to existing vocabularies and to external resources. In section 5 we address merging of the catalogues, and in section 6 the exploitation benefits that we have obtained.

3. RDFying MS data

XML is only self-descriptive about a few structural relationships: *containment*, *adjacency*, *co-occurrence*, *attribute* and *opaque reference*. Commonly, this set of relationships is not sufficient for modeling purposes. For example, the structural *containment* relation between two elements may be used to express things that are quite different semantically. Thus, often *containment* expresses *constituency* or *part-of* relation (as in *book/chapter*), but it may also express other relations, such as the ones found in *book/title* or *book/creator*. In what follows we will see the sort of problems encountered when mapping ambiguous XML elements into OWL classes and properties (Hunter and Lagoze, 2001; Tsinaraki and Christodoulakis, 2007; Xiaoshu et al., 2005).

3.1 MS: a component-based model

According to the User Manual of the META-SHARE Metadata Model[11], this model (i) is an ontology, (ii) follows the component-based mechanism as suggested by CLARIN (Broeder et al., 2008) and (iii) is formalized in a XSD, which means that all metadata instance records need to be validated against the schema.
The MS ontology includes five distinct entities: *resource*, *actor*, *project*, *document* and *license*. They are defined as sub-classes of the top entity class. Though MS is defined as ontology, the documentation does not use common ontology concepts, such as classes, instances and/or properties, but rather uses *component-based* terminology and talks in terms of *components* and *elements*. In the *component-based* approach, *elements* are used to encode specific descriptive features of the resources and are linked to conceptually similar existing elements in the Dublin Core and/or ISOcat registry. C*omponents* are complex elements and can be seen as bundle of semantically coherent *elements*.
Whereas the MS *elements* can be easily understood as properties (e.g. *name*, *gender*, *version*, etc.) and are formalized as simple-type elements in the XSD schema, *components* are more complex. MS distinguishes between 'special status components', 'linked components' and 'bare components'. The former include *person*, *organization*, *communication*, *project*, *size* and *document* and their 'special status' is meant to reflect the fact that they can be attached to various components performing different roles (i.e. *creator*, *validator*, *annotator*, etc.). 'Linked components' can be understood as relations between *components* and include concepts such as *validationTool*, *validationReport* or *validator*, among many others. Finally, 'bare components' are used to group together semantically coherent information (i.e. *metadataInfo*, *validationInfo* etc.). Formally, all components are defined as complex-type elements in the XSD schema. Semantically, 'special status components' correspond to classes whereas 'linked components' correspond to properties. As we will see in the next section, 'bare components' are harder to map.
To sum up, although the MS model is defined as being an ontology, there is no detailed (or formal) description of such an ontology that we could use. In its place, MS provides an XSD schema that 'transcodes' a component based model that implicitly formalizes an ontology.
The first thing we have to address when mapping the MS schema is the notion of *resource*. In MS, resources include corpus, lexica and tools. The schema has a root element *resource* which includes different components used to collect 'semantically related information'. Distinction between *resource* types is established at the level of a special component named *resourceComponentType*. This component is defined as a *xs:choice* between the different types of resources considered in MS. We can see the simplified tree structure in Figure 1.

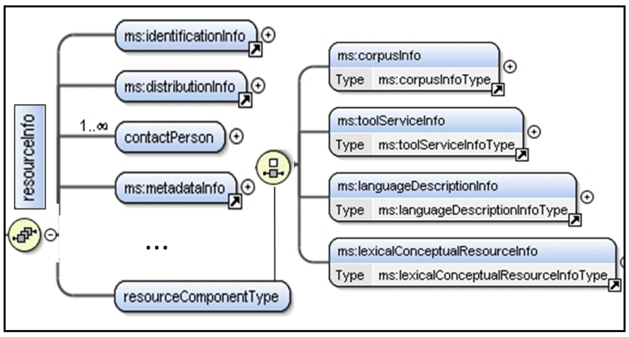

resourceInfo
ms:identificationInfo
ms:distributionInfo
1..∞
contactPerson
ms:metadataInfo
...
resourceComponentType
ms:corpusInfo
Type ms:corpusInfoType
ms:toolServiceInfo
Type ms:toolServiceInfoType
ms:languageDescriptionInfo
Type ms:languageDescriptionInfoType
ms:lexicalConceptualResourceInfo
Type ms:lexicalConceptualResourceInfoType

Figure 1: MS Resource

Resource typing can be naturally expressed in OWL. Thus, *Resource* is a class with three subclasses: *Corpus*, *lexicalConceptualResource* and *ToolService*, which in turn have subclasses. We can see the graph version in Figure 2, where continuous lines stand for subclass

[11] http://www.meta-net.eu/meta-share/META-SHARE%20%20documentationUserManual.pdf

relations and dotted lines are properties that link *Resources* with other relevant classes. Thus, for example *Resources* and *Documents* may be linked by means of ms:documentation or *ms:userManual* properties. Similarly, *Resources* and *Agents* may be linked by means of properties such as *ms:creator*, *ms:licensor*, *ms:iprRightsHolder*, etc.

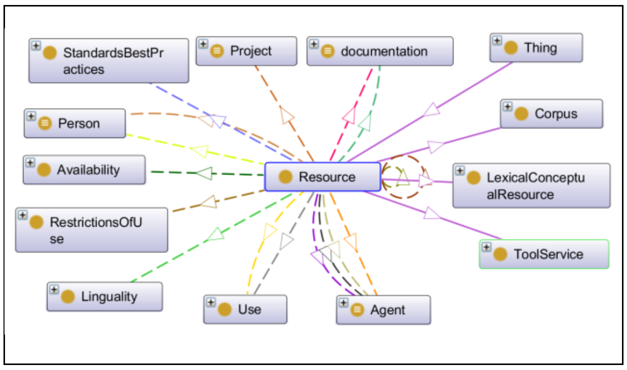


Figure 2: Resource graph

3.2 XML TO OWL

RDFying MS data implied two steps: (i) generate the ontology from the XSD schema and (ii) generate the XML instances based on decisions taken in the previous step. Table 1 shows a straightforward mapping from XSD to RDF-OWL.

XSD	OWL
xs:simpleType	rdfs:Datatype
xs:simpleType with xs:enumeration	rdfs:Datatype., plus an instance for every enumerated value.
xs:complexType	owl:Class
global element with simple type	rdfs:Datatype
local element with complex type	owl:ObjectProperty
local element with simple type	owl:DatatypeProperty

Table 1: General mapping rules from XML to RDF

However, a careful analysis of the MS schema showed that in some cases this schema was unnecessarily complex. This is partially due to the 'document-centric' approach generally followed in the MS model. Applying the rules in Table 1 to the original XSD schema would derive into a graph filled with 'superfluous' nodes. Thus, we decided to identify these nodes *before* the actual RDFication process, obtaining flatter and shallower representations. Simpler conversions derive in simpler graphs which will in turn facilitate merging and ulterior exploitation of the data. When addressing schema simplification we avoided entering into conceptual considerations and rather based our decisions only on formal aspects. As we describe in the next sections, the criteria applied take into account the tree structure of the nodes, their cardinality and the XPath axes[12].

[12] An XPath axis defines a node-set relative to the current node.

3.2.1 'Wrapping elements'

As may be derived from the rules in Table 1, any *parent/child* XML structure translates into a *Parent* class, a property such as *has_Child* and (possibly) a *Child* class. When the *Child* element is a simple element (i.e. an element that contains only text) the eventual *Child* class is omitted. When the *Child* element is a complex element, the corresponding class is created.
As we saw in Section 3.1, in some cases elements merely act as a way to organize information for human consumption. We call them 'wrapping elements'. This is the case of 'bare components' used to gather sets of relevant features. For example: in the MS schema, the root node *resourceInfo* contains a number of elements used to organize information: *identificationInfo*, *distributionInfo*, *metadataInfo*… as illustrated in Table 2.

resourceInfo/**identificationInfo(1)**/...
resourceInfo/**distributionInfo(1)**/...
resourceInfo/**contactPerson(n)**/...
resourceInfo/**metadataInfo(1)**/...
resourceInfo/**versionInfo(1)**/...
resourceInfo/**validationInfo(n)**/...
resourceInfo/**usageInfo(1)**/...
resourceInfo/**resourceDocumentationInfo(1)**/...
resourceInfo/**resourceCreationInfo(1)**/...
resourceInfo/**relationInfo(1)**/...
resourceInfo/**resourceComponentType(1)**/...

Table 2: *resurceInfo* element[13]

Assuming that (i) a class in OWL is a classification of individuals into groups which share common characteristics, and that (ii) individuals belong to some Class, we infer that for an XML element to become an OWL Class it is expected that there exist individuals belonging to this Class. In the example above, we can hardly expect individuals of *identificationInfo* type to exist. Embedded complex elements with *cardinalityMax*=1 can then safely be removed, provided they do not contain text nor attributes. This allows for a simplification of the model, as exemplified in Table 3.

resource/**identificationInfo**/resourceName
resource/**identificationInfo** /description
resource/**identificationInfo** /resourceShortName
resource/**identificationInfo** /url
…
becomes
resource/ resourceName
resource/ description
resource/ resourceShortName
resource/url
…

Table 3: Removing nodes.

Note that such a rule can be applied provided this does not derive in 'sibling conflicts'. Nodes define the scope in

[13] We use XPath expressions. Number in brackets shows node's cardinality.

which embedded elements occur and this needs to be taken into account. If we remove the *identificationInfo* element, its children nodes become children of the *resourceInfo* node. This means that *resurceName*, *description*, *resurceShortName*, etc. become sibling nodes of *contactPerson*, *validationInfo*, etc. Promoted nodes need to be unique in their new axe.

In Table 2 above, *metadataInfo* and *versionInfo* seem to be candidates for removal (they have *cardinalityMax*=1). In this case, however, such a removal will cause a 'sibling conflict': both *metadataInfo* and *versionInfo* include a *revision* element. Removing these elements would result in two sibling *revision* elements (see Table 4).

resource/**metadataInfo**/source
resource/**metadataInfo**/originalMetadataSchema
resource/**metadataInfo**/originalMetadataLink
resource/**metadataInfo**/metadataLanguageName
resource/**metadataInfo**/metadataLanguageId
resource/**metadataInfo**/metadataLastDateUpdated
resource/**metadataInfo**/**revision******
resource/**metadataInfo**/metadataCreator
resource/**versionInfo**/version
resource/**versionInfo**/**revision******
resource/**versionInfo**/lastDateUpdated
resource/**versionInfo**/updateFrequency

Table 4: 'Sibling conflict'

Naming conventions in XML design are crucial. It is recommended, as a best practice, not to name embedded elements with prefixes coming from higher nodes. Anyhow, in the MS schema, element names tend to reflect their scope. Accordingly, names of children nodes in the *metadataInfo* element follow the pattern *"*metadata*"*. However, in the case of *revision*, such a pattern does not occur.

The MS schema alternates different styles: in certain cases element and type declarations are global, whereas in other cases elements and types are declared locally. *Revision* is locally declared, and this may explain why the name used is so 'general'. It seems that elements with global types tend to generate local (i.e. context sensitive) names: i.e. the referenced type is global whereas the element's name is local. Thus, for example, we find 11 elements with type *xs:date* and they are all 'context sensitive'. Similarly, there are 19 elements with type *sizeInfoType* and, again, they are all 'context sensitive'.

These naming inconsistencies prevented us from applying more general rules and required a manual revision. Note that if naming was consistently 'context sensitive', 'wrapping nodes' could be removed without problem. On the contrary, if naming was never 'context sensitive', some prefixing rule could be applied. This illustrates how modeling with XML may make things unnecessarily complex and may derive into inconsistencies.

All in all, we identified 11 wrapping elements in the MS schema.

3.2.2 'Superfluous elements'

Besides the 'wrapping elements' described above, we identified a number of 'superfluous nodes': namely complex elements with one and only one simple element. This is, for example, what happens with the element *validationInfo/validationTool/targetResourceNameURI*. According to the general mapping rule in Table 1, such a structure would generate two classes and three properties, as shown in Table 5. In Figure 3, we show the resulting simplified graph.

Class	Property
validationInfoType	validationInfo
validationTool	validationTool
	targetResourceNameUri

Table 5: Mapping of *validationInfo*

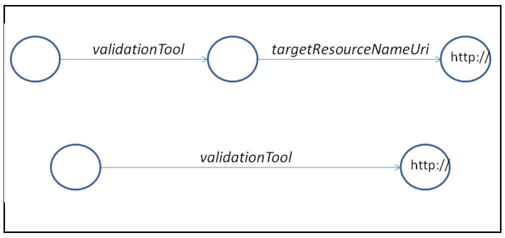


Figure 3: Removing superfluous nodes

We identified a total of 9 'superfluous nodes' in the MS schema occuring in the following contexts: *accessTool*, *annotationTool*, *creationTool*, *derivedResource*, *originalSource*, *relatedResource*, *resourceAssociatedWith*, *textNumericalContentInfo* and *validationTool*.

3.3 Enumerations

Enumerations are an illustrative example of the advantage of using RDF/OWL over XML for encoding metadata.

In XML *xs:enumeration* is not typed. Hence, we may end up having data categories spread along our XSD schemas with no explicit relation among them, or between them and external data.

In OWL, these data categories are actually ordinary resources and therefore they can be reused and linked to external vocabularies in a straightforward manner.

Simple types with *xs:enumerations* translate into an object property, a class and a corresponding instance for each enumerated value.

Thus, for example, the *lingualityType* element in Figure 4 results in the triples listed in Figure 5.

```
<xs:element name="lingualityType">
        <xs:simpleType>
                <xs:restriction base="xs:string">
                  <xs:enumeration value="monolingual"/>
                  <xs:enumeration value="bilingual"/>
                  <xs:enumeration value="multilingual"/>
                </xs:restriction>
        </xs:simpleType>
</xs:element>
```

Figure 4: Linguality in the MS schema

```
### Class
      ms:linguality
          rdf:type owl:ObjectProperty ;
          rdfs:domain ms:Resource ;
          rdfs:range ms:Linguality .
### Property
      ms:Linguality   rdf:type owl:Class .
#### Instances
      ms:monolingual a ms:Linguality .
      ms:bilingual a ms:Linguality .
      ms:multilingual a ms:Linguality .
```

Figure 5: Linguality in OWL

We found the MS schema to contain 962 *xs:enumeration* elements, of which 807 distinct ones.
As an example, the enumeration value *wordnet* occurs twice in the MS schema: one as value of the *lexicalConceptualResourceType* element and the other as value of the *compatibleLexiconType* element. In addition a *wordNet* enumeration also exists as value of the *conformanceToStandardsBestPractices* element.
In XML it is hard to organize *enumerations* while avoiding inconsistencies. The enumeration values from the elements *lexicalConceptualResourceType* and *compatibleLexiconType*, shown in Table 6, illustrate one of these inconsistencies. Notice that *mophologicalLexicon*, in the first column, is a type of lexicon. However this value is not included among the types of "lexical conceptual resources", in the second column.

compatibleLexiconType	***lexicalConceptual ResourceType***
wordnet, ***wordlist,*** *morphologicalLexicon*	***wordlist,*** *computationalLexicon,* *ontology* ***wordnet,*** *thesaurus* *framenet,* *terminologicalResource* *machineReadableDicti.* *lexicon*

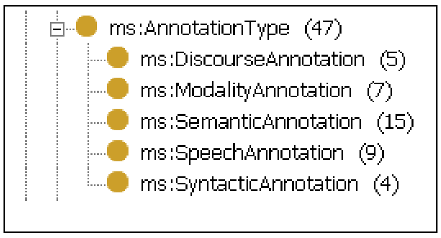
Table 6: Enumerated values for *compatibleLexiconType* and *lexicalConceptualResourceType* elements.

Sharing common vocabularies is crucial for the interoperability between data sets. Enumerations are not adequate for describing vocabularies and XML designers suggest using simple embedded elements instead of enumerations. In LMF, for example, *feature/value* assignations are performed by means of a *feat* element with two attributes (*feat* and *val*) that can be attached to almost any element. In order to be able to link 'features' and 'values' to some external vocabulary, two extra attributes are suggested: *dcr:datcat* and *dcr:valueDatcat* (Windhouwer & Wright 2012). In any case, XML is not well suited for doing this: note that *feat* declarations are local and may lead to inconsistencies and errors, not allowing for further linking, either to new ISOcat declarations or additional vocabularies.

Finally, let's mention that, in some cases, enumerations imply some sort of hierarchic organization. Since the XML machinery does not allow for 'hierarchic' enumerations, designers tend to use naming conventions. For example, the MS schema includes the *annotationType* element that encodes "*the annotation level of the resource or the annotation types a tool / service requires or produces as an output*". This element has 47 enumerations and uses prefixing to somehow 'organize' them.
Table 7 shows part of its definition, with the prefix part in bold. Table 8 shows the same example in RDF/OWL version, which allows for a better formalization.

```
... value="alignment"/>
... value="discourseAnnotation"/>
... value="discourseAnnotation-audienceReactions"/>
... value="discourseAnnotation-coreference"/>
... value="discourseAnnotation-dialogueActs"/>
... value="discourseAnnotation-discourseRelations"/>
... value="lemmatization"/>
... value="morphosyntacticAnnotation-posTagging"/>
... value="segmentation"/>
... value="semanticAnnotation"/>
... value="semanticAnnotation-certaintyLevel"/>
... value="semanticAnnotation-emotions"/>
... value="semanticAnnotation-entityMentions"/>
... value="semanticAnnotation-events"/>
```

Table 7: Enumerations of *annotationType* in XML



```
⊟ ● ms:AnnotationType (47)
     ● ms:DiscourseAnnotation (5)
     ● ms:ModalityAnnotation (7)
     ● ms:SemanticAnnotation (15)
     ● ms:SpeechAnnotation (9)
     ● ms:SyntacticAnnotation (4)
```

Table 8: Enumerations of *annotationType* in OWL

To sum up, although the general mapping rules in Table 1 can be used to generate enumerations as instances of relevant classes, this, as we have seen, requires a careful analysis of the results in order to avoid possible inconsistencies and improve on the results.

3.4 Cross references in XML

XSD allows for different design styles, going from a style where elements and types are locally declared (*Russian Doll style*) to a style where all declarations are global (*Salami Slice style*). In general, the MS schema uses global declarations, allowing for component reusability throughout the schema.
However, one of the biggest problems with the MS data is the lack of cross references among instances.Though the MS schema is full of global declarations that theoretically would facilitate component reusability, the schema does not include any ID/IDref mechanism allowing for cross reference of XML instances. All component instances are

local and cannot be referenced[14]. This poses a serious problem for the reusability (and even the usability) of the data. Consequently, searching across the resources in the MS database having the same *creator*, for example, implies using a complex element pattern matching. Similarly, uploading data into a database requires specific pattern matching validations. In contrast, in RDF/OWL, every resource is identified by an URI. Instances of *person*, *project*, *document*, etc. can be referenced everywhere, both inside the dataset as well as from external datasets.

4. Linking data

The most interesting aspect of LOD is the ability to link pieces of data, not only with data inside the same database, but also –crucially- with potentially any pieces of data out there.

4.1 Using existing vocabularies

The most basic approach to linking data (and interoperability) is to use existing vocabularies rather that creating new ones. So our first task was to identify which elements could be mapped to existing resources (properties, data categories and classes) in other well know vocabularies. We used FOAF [15], DBLP [16], DCTERMS[17], BIBO[18] and SWRC[19]. Sharing vocabularies allowed us to harmonize entities such as Person, Document and Project among the different catalogues we integrated. In Table 9 we list the correspondences between the original MS elements and the classes used in the ensuing ontology:

Document	bibo:Document , foaf:Document swrc:Document
Agent	foaf:Agent , swrc:Agent
Organization	foaf:Organizatio , swrc:Organization
Person	foaf:Person , swrc:Person
Project	foaf:Project , swrc:ResearchProject

Table 9: External vocabularies (Classes)

Similarly, some properties in the original the MS data were replaced by properties from external vocabularies. These include properties from DCTERMS, DC and FOAF vocabularies. Additionally, we used specific properties from the BioCatalogue model not included in MS. In Table 10 we list the external properties used, together with the number of triples involved in each case.

bio:demoInvocation	74
bio:endpoint	74
bio:input	51
bio:output	71
bio:serviceProvider	74
bio:serviceTechnology	146
bio:task	106
bio:wsdl	74
dc:contributor	25
dc:creator	248
dc:description	850
dc:license	1
dc:subject	16
dc:title	76
dcterms:bibliographicCitation	76
dcterms:description	5
dcterms:hasVersion	8
dcterms:isReferencedBy	35
dcterms:issued	8
dcterms:modified	8
dcterms:references	35
foaf:familyName	68
foaf:givenName	68
foaf:homepage	96
foaf:mbox	142
foaf:name	165
foaf:workplaceHomepage	66

Table 10: External properties

Finally, although the MS schema defines 'language' information (both names and codes) as simple *xs:string* type elements, we used the FAO[20] ontology for language codes (ISO638-3). Currently, our dataset includes 298 triples with language code information, using 19 different codes.

4.2 Linking to external data

Besides using existing vocabularies, we defined links between our instances and external instances whenever possible. We used the *sameAs* and *seeAlso* properties to point to the relevant records in the DBLP, the Virtual International Authority File (VIAF) and the LOC Authorities Names[21]. Table 11 lists the correspondences between *agents* and *documents* with external data.

Class	**Instances**	**sameAs**
bibo:Article	38	24
bibo:AudioVisualDocument	12	0
bibo:Book	2	0
bibo:Chapter	4	0
bibo:Report	10	1
foaf:Organization	71	63
foaf:Person	68	45
foaf:Project	26	1

Table 11: Linking *agents* and *documents*

[14] Note, in addition, that ID/IDref mechanism only guarantees cross reference inside a document, and not to external references.
[15] http://www.foaf-project.org/
[16] http://dblp.uni-trier.de/ which indexes more than 2.3 million articles and contains many links to home pages of computer scientists
[17] http://dublincore.org/documents/dcmi-terms/
[18] http://bibliontology.com/ a Bibliographic Ontology
[19] Semantic Web for Research Communities at http://ontoware.org/swrc/

[20] Food and Agriculture Organization of the United Nations (FAO) http://www.fao.org/aims/aos/languagecode.owl
[21] http://id.loc.gov/authorities/names.html

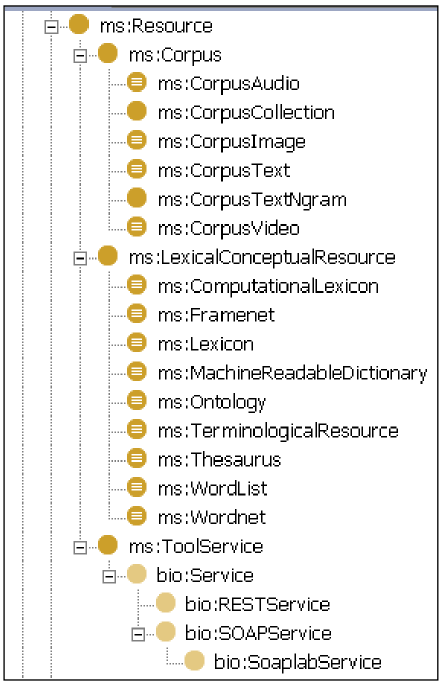
Data categories are, by large, the most prolific elements in the MS schema. As we have already mentioned, the schema includes up to 807 different enumerations that were translated into instances of relevant classes in the final model (see the example in Figure 5 above). We undertook the task of linking these instances to relevant ISOcat data categories and to DBpedia whenever possible. In Table 12 we list the data categories used: the first column indicates the class; the second column shows the number of instances for each class; and the third and fourth columns show the number of linked instances to ISOcat and DBpedia respectively (we defined a *ms:dcr* property to link to ISOcat and used *sameAs* property with DBpedia).

MS enumeration	**Num.**	**ISOcat**	**DBpedia**
AnnotationType	47	2	2
Availability	4	0	0
CharacterEncoding	140	0	4
EncodingLevel	5	5	5
LcrType	9	0	3
License	38	0	2
Linguality	3	0	0
LinguisticInformation	47	13	32
MediaType	5	4	0
ModalityType	7	0	4
MultlingualityType	3	0	0
RestrictionsOfUse	10	0	1
SegmentationLevel	16	6	8
StandardsBestPractices	44	0	18
Use	2	0	0
UseNLPSpecific	99	2	46
TOTAL	479	32	125

Table 12: Linking to ISOcat and DBpedia

5. Merging catalogues

As explained in the first section of this paper, we come from a scenario where corpus and lexicons are described in the META-SHARE node and stored in our institutional e-Repository, whereas services are registered in the BioCatalogue. In BioCatalogue, a *service* is defined as "*a container for ServiceDeployments and variants",* where a *variant* is either a *SoapService* or *RestService*, and a *ServiceDeployment* is a particular running instance of the *service*. Services also include information about testing. On the other hand, in the MS model, a service is defined as a "*form in which NLP tasks are realized and delivered to a user, without the need of acquiring and installing the corresponding tools*". Services (and tools) are *resources* which are distinguished from other resources in their *resourceComponentType*.
Due to these disparate descriptions it is not possible to directly map the information in the input *SoapServices* descriptions into the target MS *toolService* component. This implies that we need to merge the MS and BioCatalogue ontologies. The solution is to map the *variants* of a given BioCatalogue *service* as different MS *resources*, i.e. as subtypes of the MS *Service*. Table 13 shows the resulting merged model.



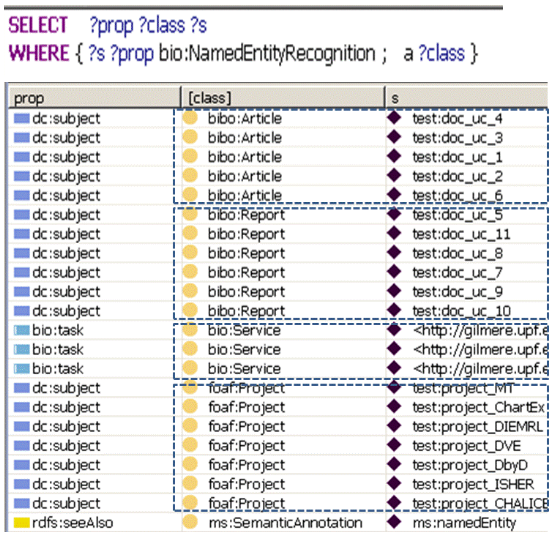
- ms:Resource
  - ms:Corpus
    - ms:CorpusAudio
    - ms:CorpusCollection
    - ms:CorpusImage
    - ms:CorpusText
    - ms:CorpusTextNgram
    - ms:CorpusVideo
  - ms:LexicalConceptualResource
    - ms:ComputationalLexicon
    - ms:Framenet
    - ms:Lexicon
    - ms:MachineReadableDictionary
    - ms:Ontology
    - ms:TerminologicalResource
    - ms:Thesaurus
    - ms:WordList
    - ms:Wordnet
  - ms:ToolService
    - bio:Service
      - bio:RESTService
      - bio:SOAPService
        - bio:SoaplabService

Table 13: MS & BioCatalogue merging

LOD is the perfect place to merge and combine data. As explained in the preceding section, by reusing data from external vocabularies we were able to re-define some MS *components* such as Person and Organization in a much more standard way (via FOAF). Similarly, for Documents, the BIBO ontology allowed us to merge our bibliographical database into the dataset. In the original MS schema, *documents* could only be attached to *resources*, either as the 'documentation' of the whole resource or as 'reports' relevant for its validation, usage, annotation and evaluation aspects. In the converted model, *documents* include articles, manuals, reports, videos, etc. and constitute an important element of the dataset. We assume that everything can be documented either directly (by means of the *ms:documentation* or *ms:validationReport* properties among others) or indirectly (by means of *dc:subject* or *dcterms:references* properties). Thus, any class and instance in the dataset may be linked to some relevant *document*. For example, the class *SOAPservice* may be referenced by some video or article. Similarly, a *Task* instance such as *namedEntityRecognition* or a *Standard* instance such as *LMF* may be also documented. We use the *dc:subject* and *dcterms:references* properties to link documents to relevant parts of the data set.

6. Exploitation benefits

Integrating disparate, yet related, datasets into a single repository has obvious benefits, especially, to the maintainability, interoperability and integrity of the data. All of this has positive consequences both on cost and functionality.
We also get an additional advantage from the properties of the RDF/OWL/SPARQL framework, which makes data exploitation simple and efficient. For example, if our user wants to know about Named Entity Recognition (NER), we can get all relevant data with a very simple query (Figure 6).



```
SELECT   ?prop ?class ?s
WHERE { ?s ?prop bio:NamedEntityRecognition ;   a ?class }
```

prop	[class]	s
dc:subject	bibo:Article	test:doc_uc_4
dc:subject	bibo:Article	test:doc_uc_3
dc:subject	bibo:Article	test:doc_uc_1
dc:subject	bibo:Article	test:doc_uc_2
dc:subject	bibo:Article	test:doc_uc_6
dc:subject	bibo:Report	test:doc_uc_5
dc:subject	bibo:Report	test:doc_uc_11
dc:subject	bibo:Report	test:doc_uc_8
dc:subject	bibo:Report	test:doc_uc_7
dc:subject	bibo:Report	test:doc_uc_9
dc:subject	bibo:Report	test:doc_uc_10
bio:task	bio:Service	<http://gilmere.upf.e
bio:task	bio:Service	<http://gilmere.upf.e
bio:task	bio:Service	<http://gilmere.upf.e
dc:subject	foaf:Project	test:project_MT
dc:subject	foaf:Project	test:project_ChartEx
dc:subject	foaf:Project	test:project_DIEMRL
dc:subject	foaf:Project	test:project_DVE
dc:subject	foaf:Project	test:project_DbyD
dc:subject	foaf:Project	test:project_ISHER
dc:subject	foaf:Project	test:project_CHALICE
rdfs:seeAlso	ms:SemanticAnnotation	ms:namedEntity

Figure 6: SPARQL query example

With this simple query we easily retrieve 'everything that has to do with NER'. In this case we get *Articles*, *Reports* and *Projects* dealing with NER as well as *Services* performing such a task. In addition, the *seeAlso* property suggests us to check *namedEntity*.

7. Data sources

All data is stored at the SPARQL endpoint: http://iula02v.upf.edu/sparql. We run a data browser located at http://lod.iula.upf.edu/types/service.

8. Conclusions

The benefits of our exercise can be summarized as follows.

On the one hand, the final RDF/OWL model is much simpler than the original XSD schema: it clearly differentiates between Classes and Properties and avoids problems typical of XML syntax, such as semantic ambiguity and order constraint. This is essential for mapping purposes. On the other hand, the open world assumption of RDF/OWL allows to naturally integrate objects from different schemas and to add further extensions, making merging of different models straightforward. We were effectively able to merge Service and Document ontologies into the MS model in a natural way.

Moreover, linking data has an additional benefit: The resulting model not only includes linking between merged catalogues but also linking to external data.

Last but not least, the RDF/OWL version of the registry (loaded in a SPARQL end point) not only provides a single unified repository but also facilitates the exploitation of the metadata records by the end-user.

The success of our experiment encourages us to apply it on a larger scenario, such as the CLARIN Component Registry. The idea behind the CMDI approach was that identifying components blocks in the original schemas would improve interoperability among models. However, the proliferation of components in the Component Registry eventually becomes a critical problem.

Conversion to RDF/OWL paves the way for more ambitious goals such as being able to derive a general ontology accounting for the different underlying schemas.

9. Acknowledgements

The work reported has been co-funded by the "Fons europeu de desenvolupament regional (FEDER), Programa operatiu FEDER de Catalunya 2007-2013, Objective 1".

10. References

Broeder, Daan, Oliver Schonefeld, Thorsten Trippel, Dieter Van Uytvanck and Andreas Witt. "A pragmatic approach to XML interoperability – the Component Metadata Infrastructure (CMDI)." Presented at Balisage: The Markup Conference 2011, Montréal, Canada, August 2 - 5, 2011. In *Proceedings of Balisage: The Markup Conference 2011. Balisage Series on Markup Technologies*, vol. 7 (2011). doi:10.4242/BalisageVol7.

Haslhofer, Bernhard and Klas, Wolfgang. (2010) A survey of techniques for achieving metadata interoperability. *ACM Comput. Surv*. 42, 2, Article 7 (March 2010), 37 pages

Hunter, Jane and Lagoze, Carl (2001) Combining RDF and XML schemas to enhance interoperability between metadata application profiles. In *Proceedings of the 10th international conference on World Wide Web* (WWW '01). ACM, New York, NY, USA, 457-466.

Lam, H. Y., Marenco, L., Shepherd, G. M., Miller, P. L., Cheung K. H. (2006) Using web ontology language to integrate heterogeneous databases in the neurosciences. AMIA ... *Annual Symposium proceedings / AMIA Symposium*. AMIA Symposium, pp. 464-468.

Tsinaraki, Chrisa and Christodoulakis, Stavros (2007) Interoperability of XML schema applications with OWL domain knowledge and semantic web tools. In *Proceedings of the 2007 OTM Confederated international conference on On the move to meaningful internet systems: CoopIS, DOA, ODBASE, GADA, and IS* - Volume Part I (OTM'07), Robert Meersman and Zahir Tari (Eds.), Vol. Part I. Springer-Verlag, Berlin, Heidelberg, 850-869.

Windhouwer, M., & Wright, S. E. (2012). Linking to linguistic data categories in ISOcat. In C. Chiarcos, S. Nordhoff, & S. Hellmann (Eds.), *Linked data in linguistics: Representing and connecting language data and language metadata* (pp. 99-107). Berlin: Springer.

Xiaoshu Wang, Robert Gorlitsky, Jonas S Almeida (2005). From XML to RDF: how semantic web technologies will change the design of 'omic' standards. *In Nature Technology*, Vol 23, No 9, pp 1099-1103, Sep 2005.