The IULA's META-SHARE LOD dataset, lessons learnt.

Editor(s): Name Surname, University, Country
Solicited review(s): Name Surname, University, Country
Open review(s): Name Surname, University, Country

Marta Villegas and Núria Bel
Institut Universitari Linguistica Aplicada. Universitat Pompeu Fabra. Roc Boronat, 138 08018 Barcelona. Spain.

Abstract. This article describes the IULA's META-SHARE LOD dataset and the RDFication task performed when moving the original XSD/XML data into RDF/OWL. The dataset has to do with language resource descriptions and it includes the LOD version of the META-SHARE model plus the IULA's language resource descriptions. The article focuses on some critical aspects when RDFying XSD/XML data. Essentially these include: the mapping of controlled vocabularies expressed in XML enumerations; the RDFication of certain unstructured data (those where unrestricted input strings may generate relevant instances) and the cleaning and linking tasks required once eventual instances are RDFied. Data cleaning and linking become crucial in a scenario where different distributed metadata nodes share their data. The eventual dataset proves efficient for data exploitation and capitalizes the efforts done. This is demonstrated by the catalog browser developed which allows retrieving relevant relations between tools and datasets, services and publications, people and projects etc.. that remained hidden in the original data and demonstrates some data mashups. This web application uses the dataset described to promote the use of language technology to researches of Humanities and Social Sciences as part of the CLARIN initiative.

Keywords: META-SHARE, RDFying data, data curation, linking data, metadata

1. Motivation

The IULA-UPF CLARIN Competence Center [1] aims to promote and support the use of technology and text analysis tools in the Humanities and Social Sciences research. The center provides specialized language technology web services as part of the European initiative CLARIN (www.clarin.eu). To promote the use of language technology, the center includes a Catalog[2] which manages, disseminates and grants access to reference information on the use of language technology projects and studies in different disciplines, especially with regard to Humanities and Social Sciences.

This Catalog has been implemented following the philosophy of Linked Open Data (LOD) and it is based on the initial LOD version of the META-SHARE (MS) model [1]. Currently, this initial version is the basis for the Linked Data for Language Technology Community Group (LD4LT) working on the Meta-Share OWL.

MS is a network of repositories of language resource (LRs), including both language data and language tools, described through a set of metadata. The MS model comprises all elements and relations assisting the description of LRs (corpora, grammars, lexica and tools) and it is formalized in an extensive XSD schema. The model includes five distinct entities: resource[3], actor, project, document and license and an exhaustive set of components that combine together to provide the full description of these core entities. MS implemented a sustainable network of repositories of language data, tools and related web services documented with high-quality metadata, aggregated in central inventories allowing for uniform search and access to resources. As MS members, we set up and maintained our own Language Re-

[1] http://www.clarin-es-lab.org/index-en.html
[2] http://lod.iula.upf.edu

[3] Language resources.

source Repository Node[4] which is synchronized with the central nodes.

The migration to the LOD framework was a natural movement, derived from the requirements of the center: the MS nodes target language technology professionals and become too technical for a potentially wider community of users. Most users are not experienced enough to cope with MS data. Thus, the original MS dataset was enriched with relevant documentation (appropriate articles, documentation, sample data and results, illustrative experiments, examples from outstanding projects, illustrative use cases, etc) to encourage our users to embrace digital tools. Crucially, the open world assumption of LOD eases data enrichment. Besides, some interesting information in the MS nodes remained invisible to end users. Additional ways to establish inferences and to maximize the inherent relations in our dataset were needed. Note finally, that MS nodes are plenty of data but remain isolated from other significant repositories. Lots of relevant information (about documents, field experts, outstanding projects, etc) is already available in external datasets and we need ways to access and exploit such amount of information.

Summarizing, we used the LOD approach (i) to maximize the information contained in our repositories, (ii) to be able to enrich the information there and (iii) to link it to external repositories and datasets.

This article describes the dataset and reports the experience when setting up a LOD catalogue. The article is organized as follows: Section 2 describes the data cleaning and linking tasks performed when moving the original dataset to the LOD version. The section focuses on: (i) the procedure and criteria followed to reuse external vocabularies; (ii) the RDFying process when addressing *xs:enumerations* (i.e. data categories), 'unstructured' data and local elements and (iii) the data enrichment procedures implemented.

Section 3 provides some metrics and links to the resources developed.

Section 4 reports the advantages of the eventual LOD dataset. These are exemplified with the Browser application developed running on the dataset.

2. RDFying: cleaning and linking

RDFying MS data implied two steps: (i) to generate the ontology from the XSD schema and (ii) to map the XML instances based on decisions taken in the previous step. When moving from SXD to RDF/OWL some general rules can be applied:

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 XSD2RDF mapping rules

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 above 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 it is described in much more detail in [2], the criteria applied take into account the tree structure of the nodes, their cardinality and the XPath axes. This allowed identifying 'removable nodes' in the original SXD schema and getting a simpler graph[5].

This section addresses some critical aspects when RDFying XSD/XML data. Essentially these include: the mapping of controlled vocabularies expressed in XML *enumerations*; the RDFying process of certain unstructured data (those where unrestricted input strings may generate relevant instances) and the cleaning and linking tasks required once eventual instances are RDFied.

[4] http://metashare.upf.edu/

[5] Removable nodes include 'Wrapping elements' and 'superfluous elements'. The former are complex elements with "cardinality max=1", they are 'document centric' in nature as they are used to group elements into conceptual dimensions (for instance to distinguish between administrative information vs. technical information). Superfluous elements are complex elements with one and only one simple element.

2.1. Controlled vocabularies: xs:enumerations

Controlled vocabularies play a critical role in metadata descriptions. XML data use *enumerations* to deal with controlled vocabularies. Thus, an enumeration limits the content of an XML element to a set of acceptable values. In XSD schemas, e*numerations* are not typed and spread along the 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 *enumerations* translate into an object property, a class and a corresponding instance for each enumerated value. The MS schema contains 962 *enumeration* elements, of which 807 distinct ones. In such a huge schema, it is not rare to find some inconsistencies. Some derive from spelling differences: for instance *word* vs. *words*, *wordlist* vs. *wordList*, etc. Other inconsistencies are not mere spelling differences; for example, grammars include an element (*compatibleLexiconType*) used to indicate the type of external lexicon that can be used with the grammar. This element has three enumerations: *wordnet*, *wordlist* and *morphologicalLexicon*. Note however, that the model distinguishes between 10 lexicon types: (namely: *wordList, computationalLexicon, ontology, wordnet, thesaurusframenet, terminologicalResource, machineReadableDictionary, lexicon* and *other)*. Not only there is a difference in spelling (*wordlist* vs *wordList*), but note that the *compatibleLexiconType* element in grammars allows for *morphologicalLexicon* which is not present in the set of admissible lexicon types. In other words: there may be a description saying "I'm compatible with a particular kind of lexicon which is not predicted in the model".

Another example is the *mediaType* element. This element includes five enumerations (*text*, *audio*, *video*, *image* and *textNumerical*). *Corpora* and *lexica* types come equipped with a *mediaType* component with a fixed value. Thus text corpora are defined with 'text' value; audio corpora are defined with 'audio' value and so on. Note, however, that *TextNgramCorpus* type is defined with 'textNgram' value: a fixed value that is not predicted in the *mediaType* element.

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 these into sub-classes.

To sum up, although the general mapping rules can be used to generate enumerations as instances of relevant classes, this requires a careful analysis of the results in order to avoid possible inconsistencies and improve on the results. In this case, such cleaning is already performed and can be reused by other MS nodes.

2.2. RDFying unstructured data

The MS *XSD* schema includes a variety of simple unrestricted elements (*ie*. those with free text content). According to general mapping rules, simple elements translate as *Datatype Properties* (properties connecting individuals with literals). In some cases, however, simple XML elements are better formalized as *Object Properties* (properties linking individuals) provided we are able to convert original textual data into relevant individuals.

The 'closed world' conception of XSD/XML framework explains the proliferation of string valued elements where a more structured representation is better suited. Schema designers need to define the closed list of *xs:enumerations*. Often, such a definition is not easy and designers decide (i) to include the so common '*other'* value to cope with unpredicted cases or, even, (ii) to use unrestricted elements instead. Note also that any change in the enumeration list (either by addition or elimination) requires a change in the schema, and this is too expensive.

Among the candidates for 'text RDFication', *language* was the most evident one. In this case, we used the Language Code Ontology[6] and defined pattern conversions that map input strings to corresponding URIS. For big datasets, OpenRefine proves very efficient in such cases: clustering allows grouping together similar results. In such cases, we recommend cleaning input string values before addressing RDFication. Just to give an example, in one of the MS central node[7] we find the following 'inconsistencies' concerning English and Spanish language codes:

[6] http://aims.fao.org/vest-registry/vocabularies/language-code-ontology-0#.VE4uEvl5Poc

[7] http://metashare.elda.org/

Value	Value counter	Resource counter
eng	518	476
en	215	174
EN	120	120
Spa	390	376
es	77	71
ES	10	10

Table 2 Language codes in MS central node

MimeType information was also a clear candidate for 'text RDFication'. We used the Internet Assigned Numbers Authority as the external vocabulary[8]. It is easy to run the SPARQL instructions like the one in Figure 1 to replace original string values by desired instances.

```
DELETE { ?s ms:mimeType "text". }
INSERT {
?s ms:mimeType http://purl.org/NET/mediatypes/text/plain>.
}
WHERE { ?s ms:mimeType "text". }
```

Figure 1 Replacing mimeType values

SPARQL queries are also very useful for identifying 'odd' values. The query in Figure 2 retrieves triples with wrong values.

```
SELECT *
WHERE {
  ?s ms:mimeType ?type
  FILTER (!regex(?type,"http://purl.org/NET/mediatypes"))
}
```

Figure 2 Identifying 'odd' values

Other text values may involve some extra work. Thus, *anotatationFormat*, *tagSet* and *theoreticalModel* could be also treated as individuals. In these cases, however, the lack of an existing vocabulary made things more complex and we did not go any further. Alternatively, we used the DBpedia and ISOcat data whenever possible.

Sometimes, string valued elements are used to 'refer' to relevant resources. Since XML/XSD is not well suited for cross-references, descriptive strings are used instead. This is case of *targetResourceNameURI* element used to collect "*The full name or a url to a resource related to the one being described…*". A manual validation is needed in this process.

Documents deserve special consideration. In our data set, documents are crucial. The enlightening nature of the catalogue led us to attach relevant documents (articles, explanatory videos, input/output samples, manuals, use cases, etc) to almost any sort of resource. Though the original MS schema only allowed documents to be attached to language resources, the eventual model allows documents to be attached to everything. In Section 2.4 we discuss the RDFying process involving documents. Here we address a small part of such a process: the *dc:subject* information in documents. *dc:subject* is used to encode the topic of the resource and in DC specifications we read: "*Typically, the subject will be represented using keywords, key phrases, or classification codes. Recommended best practice is to use a controlled vocabulary*". Though the recommendation suggests using controlled vocabularies, the fact is that free text is widely used instead. We made an effort to enrich our *document* instances with *dc:subject* properties linking to internal and external instances. The objective was to help our users to learn about the *subjects* they found in the *document* descriptions. We used the DBpedia dataset whenever possible. Thus, for example, the "IULA Treebank" article's page[9] includes links to DBpedia (and Wikipedia) for *subject* information so that the user can have a look at (sometimes rather cryptic) *subjects*.

Figure 3 "DBpedia relevant information" for *The IULA Treebank* article (http://lod.iula.upf.edu/resources/doc_24)

We wanted to use *dc:subject* information to link to external datasets (essentially the DBLP[10]). The idea was that for each article in our dataset we could get related articles in the DBLP taking the *dc:subject* as linking criteria. However, the overlapping between our *subjects* and the DBLP *subjects* was rather small and, hence, we did not implement this functionality.

2.3. Data cleaning: dealing with local XML elements

Once the mapping criteria from XSD to RDF/OWL are defined it is quite easy to set the XSLT rules to automatically RDFy XML data. Unfortunately, the process is not so straight forward and

[8] http://purl.org/NET/mediatypes/

[9] http://lod.iula.upf.edu/resources/doc_24

[10] The DBLP Computer Science Bibliography (http://www.informatik.uni-trier.de/~ley/db/)

requires data cleaning. 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[11] and cannot be referenced[12]. This poses a serious problem for the reusability (and even the usability) of the data. Note, for example, that searching across the resources in the MS database having the same *creator* implies using a complex element pattern matching. Similarly, uploading data into a database requires specific pattern matching validations.

Data cleaning was particularly critical in the case of *person*, *organization*, *project* and *document* instances. When the original data include ID and IDref attributes, these can be used to generate the corresponding URIs and the ID/IDREF mechanism guarantees data consistency. Unfortunately this is not our case: all occurrences of *persons*, *organizations*, *projects* and *documents* are locally declared and this is a potential source of problems. For *persons*, the process was quiet straight forward: the combination of the original elements *surname* and *given name* paved the way. When RDFying XML data, any *Y/X/Person* structure generates the corresponding *"Y X Person"* triple, where *Person* is a URI of the form *http://… /name_surname*. For example, the structure *Corpus_X/resourceCreator/Person* results in a triple like "Corpus*_X ms:resourceCreator John_Smith"*. The instance declaration for *John_Smith*, results of the union of all local declarations (where union removes duplicate nodes). This requires a final curation task that agrees on node values in case they are different[13]. Once, *Persons* were RDFied, we added the corresponding links to DBLP, VIAF [14]and ORCID[15] whenever possible. Table 6 shows the number of *person* individuals and the links to external datasets.

For *Organisations* and *Projects*, things were a bit more complex as naming practices for these instances are surprisingly diverse[16]. For big datasets, a preliminary version that inherits the varieties of the input XML files can be constructed straightforward. Once the data is generated, SPARQL proves very useful to identify oddities.

RDFying *Documents* was, by far, the most laborious task. MS schema defines *documentation* as an alternative choice between *documentInfo* and *documentUnstructured*. The former, consists of a complex type where all bibliographic information is split in the corresponding content element. The later, is a simple type element where the whole citation is recorded as a string. In the eventual LOD model, we need to create an instance of Bibo[17] class for each document. When the original data is a *documentInfo* element, we have a structured element that needs to be mapped into the target model. Though the mapping is quite straightforward, some critical aspects need to be addressed: in input XML data, authorship information comes as a string whereas in the target LOD model we need to identify the relevant *persons* (*creators* become instances of the *person* class). When the original data is a *documentUnstructured* element, things are much more complex. Now we have to create an instance document out of a string citation. We used DBLP and Google Scholar to search for input *unstructured* documents. This allows getting a structured bibliographic citation out of the input string. DBLP provides a RDF format which is very helpful. Google Scholar provides a BibTeX format which can be mapped to RDF.

The fact that the original XML does not allow cross-referencing (much less external-referencing) may explain why most *documentation* elements come in the unstructured shape: the effort to provide structured documentation elements does not pay off. Mapping XML documentation information into the LOD version required additional metadata curation. Though existing bibliographic catalogs can easy the task and some pattern matching rules can be defined to help in the process, we cannot avoid human work and manual validation. In any case, the effort is worth paid as not only it guarantees data quality and integrity but also benefits data exploitation (see Section 4).

2.4. Where to stop?

RDFying *documents* still implied facing a further question. We enriched our original dataset with addi-

[11] Each occurrence of a given element X is locally declared.

[12] Note, in addition, that ID/IDref mechanism only guarantees cross reference inside a document, and not to external references.

[13] It is quite common to find differences in source XML data concerning address, email, etc…

[14] Virtual International Authority File (http://viaf.org/)

[15] http://orcid.org/

[16] Note that organizations and projects may have rather long names.

[17] http://bibliontology.com/

tional documentation stuff and we end up with 155 documents with 233 different authors and this meant a lot of (hopefully avoidable) work. Since we are in the LOD model, it sounds reasonable to use external datasets whenever possible. The initial idea was to use direct references to DBLP whenever possible: whenever a document in our dataset is present in the DBLP, let's use the DBLP URI and avoid creating our 'own record' for the document.

Eventually we did not follow such an approach and rather we created 'our own records' with the corresponding *owl:sameAs* link to the DBLP. Note that we use our LOD dataset to run the Catalogue Browser. The browser requests our Virtuoso data server and displays the data as required. Having external data meant lots of connections to external servers. The first tests we performed showed that the performance was neither good nor reliable enough.

Once we decided to create 'our' *document* records, we addressed the authorship question: do we need to create 'our own' records for all the authors (ie. *dc:creators*)? At this point we took a pragmatic approach and decided to create a *person* record whenever the *person* occurs as object in some other property other than *dc:creator* (for example: *resourceCreator*, *metadataCreator*, *contactPerson*, *validator*, etc …). Thus 'prolific' *persons* have a record (with the corresponding *owl:samaAs* relation to DBLP, VIAF and/or ORCHID datasets) whereas 'odd' *persons* do not have a record and rather we use the external URI instead. Our dataset includes 155 *documents* with 290 different *creators*. Among these *creators*, 29 are locally declared, 168 are external URIS (essentially from DBLP) and 42 are encoded as string values (these are the cases where there is no external link available).

2.5. Linking

Once the data is correctly generated and cleaned it needs to be linked. This includes both internal linking (links between relevant resources in our data set) and external linking (links to external datasets). Surprisingly, our initial dataset missed some relevant internal links. This comes from the fact that the original *xsd* schema was unable to formalize certain obvious relations. For example: nothing in the source model tells us that *Named Entity Recognition* (an NLP task) has something to do with a *named entity* (an instance of 'semantic annotation'). Similarly, there is no connection between *semantic annotation* and the relevant standard *SemAF*; *semantics* and *semantic roles*; *derivation* and *morphology*, *discourse annotation* and *discourse analysis*, etc. We enriched our dataset with internal links between relevant resources. This essentially included links between *NLP tasks*, *linguistic information*, *annotation type*, *encoding level* and *standards.*

External linking includes linking of 'domain' resources (those which are part of the language technology domain) and 'general' resources (essentially: *agents*, *documents* and *projects*). For 'domain' resources, the lack of available datasets led us use DBpedia and ISOcat[18] whenever possible. For 'general' resources, we used the DBLP, ORCHID and VIAF. Table 5 and Table 6 show the number of instances linked to external dataset.

2.6. Data enrichment (data mashups)

Linked data allow the CCC Browser enriching the data available. Two procedures were defined to retrieve and display additional data. For any individual in the dataset having a *owl:sameAs* property linking to a DBpedia resource X, the Browser sends a request query to the DBpedia SPARQL endpoint asking for *subjects*[19]. For example, when browsing the 'Apertium project' page[20], the browser adds the links to DBpedia (and Wikipedia) for the *subjects* found there. Thus, the users learn that Apertium has to do with "natural language processing tools", "free software programmed in c++" and "machine translation":

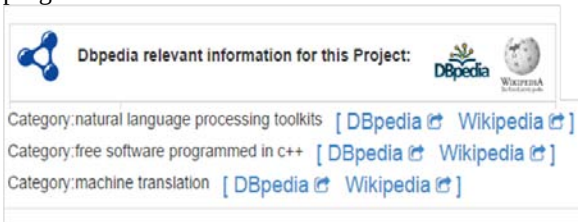

Figure 4 "DBpedia relevant information" for http://lod.iula.upf.edu/resources/project_Apertium

Persons also benefit from external data: the system sends the query to the DBLP to get all publications for a given *Person*. In this case, in order to avoid time latency and server errors, the query is not executed but rather the system displays the URI with the request[21].

[18] Data Category Registry (http://www.isocat.org/)
[19] SELECT ?subject WHERE { X dcterms:subject ?subject.}
[20] http://lod.iula.upf.edu/resources/project_Apertium
[21] For example:
http://dblp.l3s.de/d2r/snorql/?query=SELECT ?paperTitle WHERE { ?paper dc:creator

3. Data sources and metrics

The IULA's META-SHARE LOD Ontology is published at http://lodserver.iula.upf.edu/META-SHARE/ontology/. All URIS are dereferenceable both in human readable format (html) and machine readable format (turtle).

The data sources can be downloaded from https://github.com/martavillegas/metadata under the CC-BY 3.0 license.

The Virtuoso SPARQL endpoint can be accessed at http://lodserver.iula.upf.edu/sparql and the Virtuoso Faceted Browser at http://lodserver.iul.upf.edu/fct.

The CCC Browser runs on Ruby on Rails and uses the dataset described here (loaded in the Virtuoso server).

Whenever possible, the dataset uses external vocabularies. Table 2 lists the vocabularies used together with the scope of their usage.

Scope	**Vocabulary**
docs.	http://purl.org/ontology/bibo/#
licenses	http://creativecommons.org/ns#
dc	http://purl.org/dc/elements/1.1/
dcterms	http://purl.org/dc/terms/
foaf (agents)	http://xmlns.com/foaf/0.1/
Lang. codes	http://www.fao.org/aims/aos/languagecode.owl#
mime types	http://purl.org/NET/mediatypes/

Table 3 External vocabularies used

Table 4 lists some general metrics about the number of triples, classes and properties in the dataset.

Number of triples	12845
Local classes	59
External classes	13
Local object properties	35
External object properties	3
Local datatype properties	49
External datatype properties	5

Table 4 General metrics

Table 5 and Table 6 show the number of named instances grouped into 'domain' instances and 'general' instances respectively. In each case, there is the number of local instances and the number of links to external datasets. For domain instances, the number of links to external datasets is rather low and reflects the lack of available vocabularies.

<http://dblp.l3s.de/d2r/resource/authors/Mikel_L._Forcada> ; dc:title ?paperTitle . }

Domain instances	**Local**	**Dbpedia**	**ISOcat**
ms:AnnotationType	7	2	2
ms:CharacterEncoding	140	4	
ms:DiscourseAnnotation	5		
ms:EncodingLevel	5	5	5
ms:Linguality	3		
ms:LinguisticInformation	47	31	13
ms:MediaType	3		
ms:ModalityAnnotation	7		
ms:ModalityType	7	4	3
ms:MorphosyntacticAnnotation	2		
ms:MultlingualityType	3		
ms:RestrictionsOfUse	10		
ms:SegmentationLevel	16	8	6
ms:SemanticAnnotation	15		1
ms:SpeechAnnotation	9		
ms:StandardsBestPractices	44	18	
ms:SyntacticAnnotation	4		1
ms:TerminologicalResource	4		
ms:Use	1		
ms:UseNLPSpecific	100	46	
ms:UserNature	2		
bio:ServiceTechnology	5	4	
bio:Task	17	5	

Table 5 Domain instances

General instances	**Local**	**DBpedia**	**DBLP**	**VIAF**	**ORCID**
bibo:Article	95		43		
bibo:AudioVisualDoc	12				
bibo:Book	5				
bibo:Chapter	6		3		
bibo:Report	25		4		
bibo:Webpage	11				
foaf:Organization	26			7	
foaf:Person	42		25		17
foaf:Project	36	1			

Table 6 General instances

The dataset contains 215 language resources distributed into classes as listed in Table 7

Language Resources	**Local**
ms:CorpusText	22
ms:CorpusTextNgram	8
ms:ComputationalLexicon	83
ms:Ontology	3
ms:Wordnet	4
ms:TerminologicalResource	4
bio:Service	91
Total language reources	**215**

Table 7 Number of instances for language resource types

The Virtuoso faceted browser includes some sample queries which illustrate the kind of information available.

Describe META-SHARE LOD Ontology
List Tasks (label + description)
Lists lexica by language involved
Most prolific Creators
Most prolific Projects
Lists Services by Tasks

Figure 5 Sample queries in the FCT browser

4. Conclusions; making the most of data

Moving to the LOD framework allowed making the most of the data. For example, when searching for "IULA" in the a META-SHARE Central Node[22] we get 14 results which it is not much considering the node is a central one. In a relational data base it is hard to retrieve "anything that has to do with IULA" and, hence, it seems the query only gets those language resources where the string IULA occurs in de description field.

The eventual LOD dataset allows retrieving much more information in an easy way. For example, we can get all triples where the string 'IULA' occurs in the object with a simple query. Figure 6 shows the query that gives the property and the number of occurrences of 'IULA' as object of such property. Thus, we can see that 'IULA' occurs 79 times as *resource creator*, 78 times as *service provider* etc.

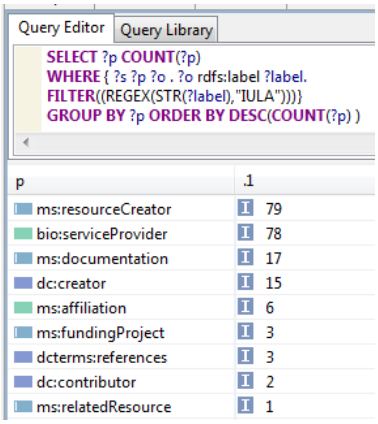

Figure 6 Searching for "IULA"

[22] http://metashare.elda.org/repository/search/?q=iula

The CCC Browser uses a similar query[23] to list all related resources together with their corresponding class (where related resources mean any resource linked to 'IULA'). Thus, for example, the 'IUA' page[24] collects this information is a box where related resources are grouped into classes (see Figure 8).

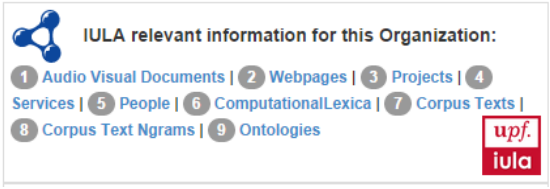

Figure 7 "IULA relevant information" for http://lod.iula.upf.edu/resources/organization_UPF-IULA

Thanks to the curation and linking tasks performed, the CCC Browser helps users to navigate throughout the dataset in a comprehensive way. For example, Figure 9 shows the "Named Entity Recognition" page[25], in this case, the system provides with: a description; the link to the *sameAs* instance in the DBpedia/Wikipedia; the links to relevant information in DBpedia (the *subjects* found in this dataset) and the links to IULA relevant information (that is, all related resources in the IULA data set). These include: semantic annotations, relevant articles and reports, projects involved in named entity recognition and services performing such a task. Taking into account that, in the original XSD schema, Named Entity Recognition was just an *xs:enumeration* value, the benefits of moving into LOD are obvious.

Data cleaning and linking become crucial in a scenario where different distributed metadata nodes share their data. Part of this curation task can be reused by other META-SHARE nodes. The eventual dataset proves efficient for data exploitation and capitalizes the efforts done. This is largely demonstrated by the catalog browser application. The catalog illustrates the potentiality of the dataset for data mashup

[23] http://lodserver.iula.upf.edu/sparql?default-graph-uri=http://MetashareLOD.org&should-sponge=&query=prefix ms: <http://lodserver.iula.upf.edu/Metashare/ontology/>
prefix test: <http://lodserver.iula.upf.edu/Metashare/resource/>
SELECT ?class ?label FROM <http://MetashareLOD.org>
WHERE { ?s ?p test:organization_UPF-IULA ; a ?class ; rdfs:label ?label.
FILTER (!regex(?class,"NamedIndividual"))}
GROUP BY ?class ORDER BY ?class&format=text/html&debug=on&timeout=

[24] http://lod.iula.upf.edu/resources/organization_UPF-IULA

[25] http://lod.iula.upf.edu/resources/NamedEntityRecognition

using data from different sources (DBpedia and DBLP).

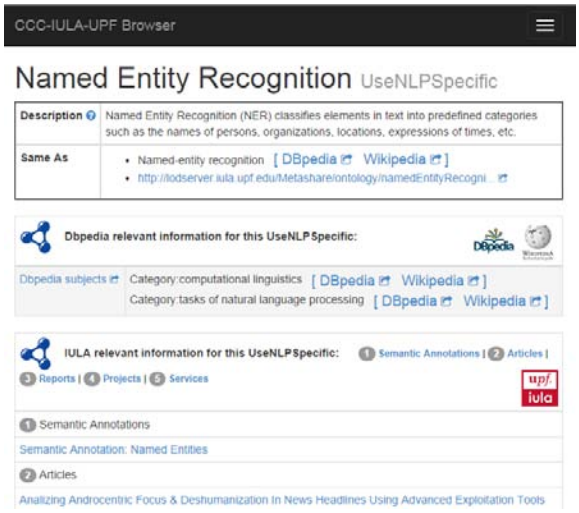

Figure 8 Named Entity Recognition at the CCC Browser (http://lod.iula.upf.edu/resources/NamedEntityRecognition)

5. Acknowledgements

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

References

[1] Maria Gavrilidou, Penny Labropoulou, Elina Desipri, Stelios Piperidis, Harris Papageorgiou, Monica Monachini, Francesca Frontini, Thierry Declerck, Gil Francopoulo, Victoria Arranz, and Valérie Mapelli. The META-SHARE Metadata Schema for the Description of Language Resources. Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC-2012), Istanbul, Turkey, May 23-25, 2012

[2] Marta Villegas, Maite Melero and Núia Bel. Metadata as Linked Open Data: mapping disparate XML metadata registries into one RDF/OWL registry. Proceedings of the Nine International Conference on Language Resources and Evaluation (LREC-2014), Reykjavik, Iceland, May 23-25, 2014

[3] 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.