Jason Grey

LIS 882 - Metadata for Digital Resources

November 12, 2021

Metadata Schema Report

OWL - The Web Ontology Language

Introduction

OWL (Web Ontology Language) is a language developed and maintained by the W3 consortium for the purpose of creating formal taxonomies primarily intended to be used by artificial intelligence agents. It is in many ways a descendant of the RDF specification and uses RDF as part of its core group of elements. Version 1 of OWL was published in 2004, and OWL 2, which has several new features but is backwards compatible with OWL, was published in 2009. OWL extends the idea of the "Semantic Web" that RDF was designed to facilitate by applying a consistent standard to ontologies within a specific domain of knowledge, so that they can be combined and shared. OWL differs from RDF in its expressiveness and logical completeness, which allows for more sophisticated algorithmic queries. OWL both extends and depends on the formal semantics and namespaces of RDF and RDF Schema.

While the purpose of OWL ontologies is to represent things in the real world, this kind of logical representation is somewhat different conceptually than a metadata standard such as MODS or VRA. The most important feature of OWL is Classes, which are abstract labels intended to represent both things in the real world, and the relationships between those things. In a metadata standard such as MODS, relationships are mostly suggested or implied by similarity of certain elements; it is assumed that all works that have the element <form>painting</form> have something in common, and if a controlled vocabulary is used for values, then there is an established definition of what that commonality means. In OWL, by contrast, relationships between Classes are specified as part of the Class definition; there can also be relationships between Class members that are implied by the properties of their elements. By encoding these

class-based relationships into each ontology that is constructed, it is possible to create complex semantic relationships between individual items that can be processed by machine agents and used to answer structured queries, which in many cases are constructed from natural language questions. There are different versions of OWL (known as sublanguages in OWL 1 and profiles in OWL 2) which differ primarily by the level of logical expressiveness and complexity allowed in their class definitions.

Core principles and examples of syntax

The fundamental component of an OWL ontology is a set of Classes, each of which represents a type of entity. Classes are hierarchical, and the root element of all Classes is "Thing"; any class that is defined without a parent is assumed to be a child of "Thing". A user defined class will have an ID attribute, defined by the RDF syntax. In XML, defining a class of Animals looks like:

```
<owl:Class rdf:ID="Animal"></owl:class>
```

By convention, class definitions are singular nouns. The owl: and rdf: namespaces used here would be defined in the XML header. There are several other syntaxes available to encode OWL definitions; XML is used here because it is the most verbose and easiest to read for beginners.

A class as broad as "Animal" is not very useful without subclasses. A parent class can be assigned to a newly created class:

```
<owl:Class rdf:ID="Horse">
     <rdfs:subClassOf rdf:resource="#Animal" />
</owl:class>
```

Note that the imported RDF syntax rules are used again, as well as RDF Schema. Now that the class is defined at our desired level of specificity, individuals can be added to the class:

```
<Horse rdf:ID="Secretariat" />
<Horse rdf:ID="Bucephalus" />
```

Additional properties and relationships can be added to both Classes and Individuals. There are two types of properties: Datatype properties are similar to properties in a metadata schema such as MODS, where a value is attached to an entity of some sort. For instance, to assert that a Horse has 4 legs:

```
<owl:datatypeProperty rdf:ID=”numberOfLegs>
      <rdfs:domain rdf:resource=”#Animal” />
      <rdfs:range rdf:resource=”&xsd;positiveInteger” />
</owl:datatypeProperty>
<Horse>
      <NumberOfLegs>4</numberOfLegs>
</Horse>
```

This defines a property called numberOfLegs, that is a property of Animals, has a value that is a positive integer, and then that Horses have 4 “numberofLegs”.

Object properties define relationships between classes or individuals. A property called ancestorOf would be defined like this:

```
<owl:objectProperty rdf:ID=”ancestorOf>
      <rdfs:domain rdf:resource=”#Animal” />
      <rdfs:range rdf:resource=#Animal” />
</owl:datatypeProperty>
<Bucephalus>
      <AncestorOf rdf:resource=”#Secretariat” />
</Bucephelus>
```

Properties have a variety of restrictions and characteristics that can be applied to them. For instance, hasAncestor could be made a transitive property, stating that if x is the ancestor of y, and y is the ancestor of z, then z is the ancestor of x. Many other common logical relationships can be applied to properties, which can then be used to find relationships between individuals via structured queries. There are also set operators that can be used to create relationships between classes; the disjoint operator, for instance, indicates classes that are mutually exclusive of each other.

One of the main distinctions between the simpler sublanguages (OWL Lite and OWL DL) and OWL Full is that in OWL Full, an Individual can have the same properties and relationships as a Class. This allows complex logical associations to be made that can be especially useful when combining multiple ontologies. An Individual within one dataset can be treated as a Class when querying a different dataset. As an example, a genealogy website might consist largely of People, defined by their relationship to each other. Using a medical ontology, each Person could be the parent of a Diseases class, making it possible to ask questions like “does this branch of this family have a high rate of this particular Disease?”

Resources

OWL 1 Language Guide, 2004 : https://www.w3.org/TR/owl-guide/ : this is the home page of the specifications for version 1 of OWL, it has an overview of the language and links to the complete reference.

OWL 2 Language Guide, 2012 : https://www.w3.org/TR/2012/REC-owl2-primer-20121211/ : home page of version 2, describes the changes in new version and has a primer intended for those new to the language in general

OWL @ Manchester : http://owl.cs.manchester.ac.uk/ : The University of Manchester has been one of the most active research centers for OWL in the past, although this project seems to have come to a halt in 2017.

OWL Ontologies list : https://protegewiki.stanford.edu/wiki/Protege_Ontology_Library : a list of ontologies built in OWL maintained by Stanford University, this provides an idea of the kinds of projects that ontologies have been developed for, although the majority of the links on this list are broken.

Conclusion and future directions

As can be seen by the poorly maintained state of the resources listed here, development and practical use of OWL ontologies is not currently very robust. It is difficult to identify a precise cause for this lack of interest in the technology, given the level of interest in linked data and artificial intelligence that OWL was designed to be useful for, but Duca (2021) and others suggest that the focus of OWL on structured data may be working against its widespread use, as machine learning techniques that process unstructured data become more powerful and productive. It is impossible to predict the direction of technological development, though, so it is entirely possible that the OWL languages will experience a resurgence of interest at some point.

PBCore - Public Broadcasting Core Metadata Dictionary

Introduction

Development on PBCore was begun in 2001 by the Corporation for Public Broadcasting; it is currently maintained by the American Archive of Public Broadcasting. It is a good example of a metadata standard that has been developed for use within a specific industry; the schema is intended primarily for the cataloging and archiving of audiovisual media, and is especially focused on broadcast media and having elements useful to broadcasters. Version 1 of PBCore was released to the public in 2005, and the current standard PBCore 2.1 was released in 2015. Because version 2 is not fully backwards-compatible, there are still some institutions using PBCore 1. It is fully open source and free to use. It is intended to be encoded in XML and the .xsd file to validate the schema is available on their website. While the pbCore schema was developed specifically for public broadcasters, to make it easier for them to share resources with each other, it has been adopted by some media entities and by some content management companies outside of the public broadcasting world as well, who have found it useful to have a common vocabulary specifically made for communicating with radio and TV producers.

Core principles and examples of syntax

Like VRA, PBCore has three root-level elements that are the basis for different types of records - pbCoreCollection, pbCoreDescriptionDocument, and pbCoreInstantiationDocument - the meanings and structure of these are on the surface similar to the top level VRA elements Collection, Work, and Image, but there are important differences in their conception and use. pbCoreCollection is a container for a group of documents; its only valid subelement is one or more pbCoreDescriptionDocument elements. pbCoreDescriptionDocument is analogous to the

VRA ‘Work’ element; it represents a piece of intellectual content that can (and usually does) have one or a series of instantiations, which are subelements of the type pbCoreInstantiation. The definition of an ‘instantiation’ is deliberately left flexible, but its most common use is to refer to physical or virtual objects such as videotapes or computer files that are a functional “representation of the asset”. So unlike VRA, most of the “copies” or “images” of a work are contained within a single XML record, rather than being a separate record referred to by the Relation element. The third root element, pbCoreInstantiationDocument, is used to create a record of a single instantiation; according to the pbCore documentation, “This is most commonly used when referenced from other schemas”. This element is primarily used for flexibility, particularly when importing or exporting metadata to or from other schemas that have a different method of hierarchical organization than pbCore. Many operational uses of pbCore would not have any need for it.

Within pbCoreDescriptionDocument, the main element to describe “things”, the required elements are pbCoreIdentifier, pbCoreTitle, and pbCoreDescription. pbCoreIdentifier is critical because the schema is designed to be used as part of a working database, which requires a way to uniquely identify each object in the database, and a title and description are needed for human useability. That pbCoreDescription is required is another difference from more general-purpose schemas like VRA or Dublin Core; pbCore has a greater focus on searchability and readability of records, because it was created with production environments like TV stations in mind. The use case scenarios for pbCore applications often involve a producer or creator looking for footage or audio on a specific topic to be incorporated into another project, or looking up the date or other metadata about an old project; given the variety of subject matter that could possibly be covered, verbose text descriptions are more useful for searchers then subject headings or other controlled

vocabularies. (There is another element, pbCoreSubject, which is used for controlled vocabularies like LCSH.)

There are 17 optional subelements for pbCoreDescriptionDocument, many of which are common to most metadata schemas (pbCoreCreator, pbCoreRelation). The most significant is PbCoreInstatiation, which as noted is the wrapper element for any physical or digital manifestations of the work. It is here where we find elements that are most specifically intended for audiovisual materials, like instantiationTracks, instantiationColors, or instantiationDataRate (note the pbCore prefix goes away for these elements). There are also more familiar elements like instantiationDate and instantiationLocation. A few of the subelements of in the 'instantiation' space have controlled vocabularies provided by pbCore to cover specific technical aspects of audiovisual material, notably instantiationPhysical.

pbCore also provides extension subelements for both the pbCorDocumentDescription and pbCoreInstantiation elements. One of the primary intended uses of these is to facilitate migration and sharing of data between systems that use other metadata schemas.

Resources

PbCore Main Website : https://pbcore.org/ : This is the main hub for resources for pbCore. It has a brief history of the development and rationale for creating pbCore, tutorials and a complete guide to the schema, links to associated resources like crosswalks for other schemas and their in-house cataloging software, and a list of organizations that are using pbCore.

American Archive of Public Broadcasting : https://americanarchive.org/ : This is probably the largest public project that uses pbCore as its database system; it doesn't really "show off" the underlying system but gives a good idea of the kind of content that pbCore is intended to manage.

Omeka Plugin for pbCore : https://omeka.org/classic/docs/Plugins/PBCore/ : this would be a good way to do a small project using pbCore, for someone who wanted to dip a toe in the water

Conclusion and future directions

The pbCore schema is relatively new, and the website lists a dozen organizations that actively use it in their work. Given the focus on audiovisual content, and the many features of the schema that are specifically designed to enable the work of producers and creators in these media, it seems likely that there are many other types of organizations that would benefit from pbCore's standardized metadata, particularly the ability to easily share and search medium-specific information such as running length and video resolution.

WORKS CITED

Duca, Angelica L. (2021, July 18) *Is the Semantic Web Really Dead?* Geek Culture, https://medium.com/geekculture/is-the-semantic-web-really-dead-7113cfd1f573

Teixeira, F. P. (2016, October 25) *OWL is Dead, Long Live OWL!* Vaticle. https://medium.com/vaticle/owl-is-dead-long-live-owl-7a93f8bdea59