00 - Title page

Linked Archival Metadata: A Guidebook

draft Draft Draft! Draft!!! DRAFT!!! D R A F T ! ! !

--
LiAM & Eric L. Morgan
http://sites.tufts.edu/liam/

April 8, 2014

1. Executive Summary

[The Executive Summary will list core objectives, anticipated outcomes, and implications that will provide administrators or other senior leaders with the information that they will need in order to understand the benefits and potential costs of this path.]

Introduction

> "Let's go to Rome!"

The purpose of this guidebook is to describe in detail what linked data is, why it is important, how you can publish it to the Web, how you can take advantage of your linked data, and how you can exploit the linked data of others. For the archivist, linked data is about universally making accessible and repurposing sets of facts about you and your collections. As you publish these fact you will be able to maintain a more flexible Web presence as well as a Web presence that is richer, more complete, and better integrated with complementary collections. The principles behind linked data are inextricably bound to the inner workings of the Web. It is standard of practice that will last as long as the Web lasts, and that will be long into the foreseeable future. The process of publishing linked data in no way changes the what of archival practice, but it does shift how archival practice is accomplished. Linked data enables you to tell the stories of your collections, and it does in a way enabling you to reach much wider and more global audiences.

Linked Archival Metadata: A Guidebook provides archivists with an overview of the current linked data landscape, define basic concepts, identify practical strategies for adoption, and emphasize the tangible payoffs for archives implementing linked data. It focuses on clarifying why archives and archival users can benefit from linked data and will identify a graduated approach to applying linked data methods to archival description.

"Let's go to Rome!" This book uses the metaphor of a guidebook, specifically a guidebook describing a trip to Rome. Any trip abroad requires an understanding of the place where you intend to go, some planning, some budgeting, and some room for adventure. You will need to know how you are going to get there, where you are going to stay, what you are going to eat, what you want to see and do once you arrive, and what you might want to bring back as souvenirs. You behooves you to know a bit of the history, customs, and language. This is also true of your adventure to Linked Data Land. To that end, this guidebook is divided into the following sections:

* What is linked data and why should I care? - A description of the beauty of Rome and why you should want to go there
* Linked data: A Primer - A history of Rome, an outline of its culture, its language, its food, its customs, and what it is all about today
* Strategies - Sets of things you ought to be aware of and plan for before embarking on your adventure
* Doing the work - Itineraries for immersing yourself and enjoying everything the Eternal City has to offer once you have arrived
* Details - Lists of where to eat, what to do depending on your particular interests, and how to get around the city in case you need help

The Guidebook is a product of the Linked Archival Metadata planning project (LiAM), led by the Digital Collections and Archives at Tufts University and funded by the Institute of Museum and Library Services (IMLS). LiAM’s goals include defining use cases for linked data in archives and providing a roadmap to describe options for archivists intending to share their description using linked data techniques.

What is linked data, and why should I care?

> "Tell me about Rome. Why should I go there?"

Linked data is a standardized process for sharing and using information on the World Wide Web. Since the process of linked data is woven into the very fabric of the way the Web operates, it is standardized and will be applicable as long as the Web is applicable. The process of linked data is domain agnostic meaning its scope is equally apropos to archives, businesses, governments, etc. Everybody can and everybody is equally invited to participate. Linked data is application independent. As long as your computer is on the Internet and knows about the World Wide Web, then it can take advantage of linked data.

Linked data is about sharing and using information (not mere data but data put into context). This information takes the form of simple "sentences" which are intended to be literally linked together to communicate knowledge. The form of linked data is similar to the forms of human language, and like human languages, linked data is expressive, nuanced, dynamic, and exact all at once. Because of its atomistic nature, linked data simultaneously simplifies and transcends previous information containers. It reduces the need for profession-specific data structures, but at the same time it does not negate their utility. This makes it easy for you to give your information away, and for you to use other people's information.

The benefits of linked data boil down to two things: 1) it makes information more accessible to both people as well as computers, and 2) it opens the doors to the creation any number of knowledge services limited only by the power of human imagination. Because it standardized, agnostic, independent, and mimics human expression linked data is more universal than many of the current processes of information dissemination. Universality infers decentralization, and decentralization promotes dissemination. On the Internet anybody can say anything at anytime. In the aggregate, this is a good thing and it enables information to be combined in ways yet to be imagined. Publishing information as linked data enables you to seamlessly enhance your own knowledge services as well as simultaneously enhance the knowledge of others.

> "Rome is the Eternal City. After visting Rome you will be
> better equipped to participate in the global conversation of
> the human condition."

Linked data: A Primer

> "Okay. My interest is piqued. Please tell me more about Rome's history, culture, and geography. What are the people like, what do they do for a living, and how can I get around if I'm going to stay for a while?".

This section describes linked data in much greater detail. Specifically, this section introduces the reader to the history of linked data, a data model called RDF, RDF "serializations", RDF publishing models, ways RDF has been used, SPARQL,...

History

> "Rome rose, fell, and has risen again."

The history of linked data begins with a canonical Scientific American article, "The Semantic Web" by Tim Berners-Lee, James Hendler, and Ora Lassila in 2001. [1] The article described an environment where Internet-wide information was freely available for both people and computers with the ultimate purpose of bringing new knowledge to light. To do this people were expected to: 1) employ a data model called RDF for organizing their information, and 2) express the RDF as XML files on the Web. In a time when the predominate mode of data modeling was rooted in relational databases, the idea of RDF was difficult for many people to understand. Of the people who did understand, many them thought expressing RDF as XML made things too difficult to read. Despite these limitations, there was a flurry of academic research done against the idea of the Semantic Web, but the term "linked data" had yet to be coined.

Around this same time REST-ful computing was being articulated by the Internet community. Simply put, REST-ful computing is a way for computers to request and share information over the World Wide Web. All one usually had to do is submit a very long URL (complete with numerous name/value pairs) to a Web server, and the Web server was then expected to return computer readable data. Many computer programmers and people who could write HTML picked up on the idea quickly. REST-ful computing was seen as immediately practical with little need to learn anything about "data models". Because the ideas behind the Semantic Web may have been oversold and because REST-ful computing was seen as so easy to implement, REST-ful computing flourished (and continues to flourish) while interest in the Semantic Web waned.

Then, in 2006, Tim Berners-Lee concretely described how to make the Semantic Web a reality. [2] In it he listed a four-step process for making content freely available on the Web. It is a practical process many people can identify with. The description also advocated for simpler URLs (URLs sans long name/value pairs) for identifying anything in the world -- people, places, or things both real and imaginary. At this same time new ways of expressing RDF were articulated and becoming popular. RDF manifested as XML was no longer the only choice. Also at this same time a few entrepreneurial individuals were beginning to provide software applications and services for creating, maintaining, and distributing RDF. This made the work of some people easier. An increasing number of specialized communities -- governments, scientific communities, professional associations, and "cultural heritage institutions" -- began making their data and metadata freely available on the Web, accessible via REST-ful computing or RDF. These developments, plus a kinship of "all things open" (open source software, open access publishing, open data, etc.) to the ideals of the Semantic Web, probably contributed to the current interest in the newly coined phrase "linked data". Progress and developments in linked data (now rarely called the Semantic Web) continue but at a more measured pace. Linked data communities are strengthening. The ideas behind modeling data as RDF are becoming better understood. Production-level RDF services are being implemented. While the ideas behind RDF, linked data, and ultimately the Semantic Web have yet to become mainstream, interest is now in waxing phase.

What is RDF?

"Rome is inextricably linked to its Roman heritage."

Linked Data is a process for sharing human knowledge on the World Wide Web. It is about asserting relationships between things and then linking these things together to express knowledge. These two things (asserting relationships and linking) are at the very heart of linked data, and they are the defining characteristics of RDF. This section describes RDF in greater detail.

RDF is an acronym for Resource Description Framework. As the name implies, it is a structure (framework) for asserting relationships (descriptions) between things (resources). It is a model for organizing data. Unlike the data model of spreadsheets made up of rows & columns, or the data model of joined tables as in relational databases, the data model of RDF is based on the idea of a triple -- a simple "sentence" with three distinct parts: 1) a subject, 2) a predicate, and 3) an object. The subject of each triple is expected to be a URI (for the time being, think "URL"), and this URI is expected to point to things either real or imaginary. Similarly, the object of each triple is a URI, but it can also be a literal -- meaning a word, phrase, narrative, or number. Predicates take the form of URIs too, and they are intended to denote relationships between the subjects and objects. To extend the analogy of the sentence further, think of subjects and objects as if they were nouns, and think of predicates as if they were verbs.

RDF statements are often illustrated as arced graphs where subjects and objects are nodes in the illustration and predicates are lines connecting the nodes:

```
  [ subject ] --- predicate ---> [ object ]
```

The "linking" in linked data happens when sets of RDF statements share common URIs. By doing so, the subjects of RDF statements end up having many characteristics, and the objects of URIs point to other subjects in other RDF statements. This linking process transforms independent sets of RDF statements into a web of interconnections, and this is where the Semantic Web gets its name:

```
                / --- a predicate ---------> [ an object ]
  [ subject ] - |  --- another predicate ---> [ another object ]
                \ --- a third predicate ---> [ a third object ]

                                                    |
                                                    |
                                           yet another predicate
                                                    |
                                                    |
                                                   \ /

                                                [ yet another object ]
```

An example is in order. Suppose there is a thing called Rome, and it will be represented with the following URI: http://example.org/rome. We can now begin to describe Rome using triples:

subjects	predicates	objects
http://example.org/rome	has name	"Rome"
http://example.org/rome	has founding date	"1000 BC"
http://example.org/rome	has description	"A long long time ago,..."
http://example.org/rome	is a type of	http://example.org/city
http://example.org/rome	is a sub-part of	http://example.org/italy

The corresponding arced graph would look like this:

```
                                  / --- has name ------------> [ "Rome" ]
                                 |  --- has description -----> [ "A long time ago..." ]
```

```
  [ http://example.org/rome ] - |  --- has founding date ---> [ "1000 BC" ]
                                |  --- is a sub-part of  ---> [ http://example.org/italy ]
                                 \ --- is a type of --------> [ http://example.org/city ]
```

In turn, the URI http://example.org/italy might have a number of relationships asserted against it also:

subjects	predicates	objects
http://example.org/italy	has name	"Italy"
http://example.org/italy	has founding date	"1923 AD"
http://example.org/italy	is a type of	http://example.org/country
http://example.org/italy	is a sub-part of	http://example.org/europe

Now suppose there were things called Paris, London, and New York. They can be represented in RDF as well:

subjects	predicates	objects
http://example.org/paris	has name	"Paris"
http://example.org/paris	has founding date	"100 BC"
http://example.org/paris	has description	"You see, there's this tower..."
http://example.org/paris	is a type of	http://example.org/city
http://example.org/paris	is a sub-part of	http://example.org/france
http://example.org/london	has name	"London"
http://example.org/london	has description	"They drink warm beer here."
http://example.org/london	has founding date	"100 BC"
http://example.org/london	is a type of	http://example.org/city
http://example.org/london	is a sub-part of	http://example.org/england
http://example.org/newyork	has founding date	"1640 AD"
http://example.org/newyork	has name	"New York"
http://example.org/newyork	has description	"It is a place that never sleeps."
http://example.org/newyork	is a type of	http://example.org/city
http://example.org/newyork	is a sub-part of	http://example.org/unitedstates

Furthermore, each of "countries" can be have relationships denoted against them:

subjects	predicates	objects
http://example.org/unitedstates	has name	"United States"
http://example.org/unitedstates	has founding date	"1776 AD"
http://example.org/unitedstates	is a type of	http://example.org/country
http://example.org/unitedstates	is a sub-part of	http://example.org/northamerica
http://example.org/england	has name	"England"
http://example.org/england	has founding date	"1066 AD"
http://example.org/england	is a type of	http://example.org/country
http://example.org/england	is a sub-part of	http://example.org/europe
http://example.org/france	has name	"France"
http://example.org/france	has founding date	"900 AD"
http://example.org/france	is a type of	http://example.org/country
http://example.org/france	is a sub-part of	http://example.org/europe

The resulting arced graph of all these triples might look like this:

  [INERT ARCED GRAPH HERE.]

From this graph, new information can be inferred as long as one is able to trace connections from one node to another node through one or more arcs. For example, using the arced graph above, questions such as the following can be asked and answered:

  * What things are denoted as types of cities, and what are their names?
  * What is the oldest city?
  * What cities were founded after the year 1 AD?
  * What countries are sub-parts of Europe?
  * How would you describe Rome?

In summary, RDF is data model -- a method for organizing discrete facts into a coherent information system. The model is built on the idea of triples whose parts are URIs or literals. Through the liberal reuse of URIs in and between sets of triples, questions surrounding the information can be answered and new information can be inferred. RDF is the what of linked data. Everything else (ontologies & vocabularies, URIs, RDF "serializations" like RDF/XML, triple stores, SPARQL, etc.) are the how's. None of them will make any sense unless the reader understands that RDF is about establishing relationships between data for the purposes of sharing information.

Ontologies & vocabularies

  "What languages do they speak in Rome?"

If RDF is built on "sentences" called triples, then by analogy ontologies & vocabularies are the "languages" of RDF. This section describes the role of ontologies & vocabularies in linked data.

Linked data is about putting data in the form of RDF on the Web. RDF is a data model for describing things (resources), and it made up of three parts: 1) subjects, 2) predicates, and 3) objects. The things being described are the subjects of RDF triples. They represent the things you own. The combined use of predicates and objects form the descriptions of the resources. These descriptions are akin to a language, or in the parlance of RDF, they are ontologies & vocabularies. While it is perfectly fine to create your own language to describe your own things, it behooves you to use one or more ontologies & vocabularies of others. Otherwise your descriptions will exist in a virtual silo with no interaction with outside resources. When outside ontologies & vocabularies are not employed in RDF, then the purpose of the linking in linked data gets defeated.

RDF ontologies & vocabularies are comprised of classes of objects and properties of those objects. The classes of objects posit the existence of things. They might posit the class of all people, places, events, etc. Properties are characteristics of the classes. People might have names and birth dates. Places have geographic coordinates. Events have dates, times, and descriptions.

There are quite a number of existing ontologies & vocabularies. Some of them of interest to readers of this guide are listed in another section, but a few are briefly discussed here. The first is FOAF (Friend Of A Friend). [3]. This ontology/vocabulary is used to describe people. It defines a number of classes, including but not limited to: agent, person, and document. Agents have properties such as mboxes (email addresses), various identifiers, topics of interest, etc. A person, which is a subclass of Agents inherits all of the properties of an agent, but also has a few of its own such as family names, various types of home pages, and images. Documents have properties such as topics and primary interests. If the resources you were describing were people (or agents), then it might want to draw on the FOAF ontology/vocabulary. If the entity named Rome had an email address, then its RDF arced graph might look like this:

  [http://example.org/rome] --- has mbox ---> [mailto:info@rome.it]

Another ontology/vocabulary is DCMI Metadata Terms ("Dublin Core"). [4] It posits the existence of things like: agent, rights statement, standard, and physical resource. It includes properties such as creator, title, description, various types of dates, etc. While the "Dublin Core" standard was originally used to describe bibliographic materials, it has matured and been widely adopted by a growing number of ontologists.

In a project called Linking Lives at the Archives Hub, an ontology/vocabulary for archival description was

created. [5, 6] This ontology includes a few classes from FOAF (document, agent, etc.) but also has classes such as repository, archival resource, biographical history, and finding aid. Properties include various dates, extent, title, or has biographical history.

A final example is VIAF (Virtual International Authority File). [7] This "ontology" is more akin to a "controlled vocabulary", and it is a list of people's names and URIs associated with them. VIAF is intended to be used in conjunction with things like DCMI's creator property. For example, if Romulus, one of the mythical twins and founders of Rome were associated with the entity of Rome, then the resulting arced graph might look like this:

```
  [ http://example.org/rome ] --- has creator ---> [ http://viaf.org/viaf/231063554/ ]
```

There are other controlled vocabularies of interest to the readers of this book, including additional name authority files, subject headings, language codes, country listings, etc. These and other ontologies & vocabularies are listed later in the guidebook.

RDF and linked data is about making relationships between things. These relationships are denoted in the predicates of RDF triples, and the types of relationships are defined in ontologies & vocabularies. These ontologies & vocabularies are sometimes called schema. In the world of bibliography (think "Dublin Core"), these relationship types include things such "has title", "has subject", or "has author". In other ontologies, such as Friend of a Friend (FOAF), there are relationship types such as "has home page", "has email address", or "has name". Obviously there are similarities between things like "has author" and "has name", and consequently there are other ontologies (schemas) allowing equivocation, similarity, or hierarchy to be denoted, specifically RDFS, SKOS, and OWL.

In the world of archives, collections and their items are described. Think metadata. Some of this metadata comes from name authority lists and controlled vocabulary terms. Many of the authority lists and controlled vocabulary terms used by archives exist as linked data. Thus, when implementing RDF in archives it is expected to state things such as "This particular item was authored by this particular URI", or "This particular collection has a subject of this particular URI" where the URIs are values pointing to items in named authority lists or controlled vocabularies.

Probably one of the more difficult intellectual tasks you will have when it comes to making your content available as linked data will be the selection of one or more ontologies used to make your RDF. Probably the easiest -- but not the most precise -- way to think about ontologies is as if they were fields in a MARC record or an EAD file. Such an analogy is useful, but not 100% correct. Probably the best way to think of the ontologies is as if they were verbs in a sentence denoting relationships between things – subjects and objects.

But if ontologies are sets of "verbs", then they are akin to human language, and human language is ambiguous. Therein lies the difficulty with ontologies. There is no "right" way to implement them. Instead, there is only best or common practice. There are no hard and fast rules. Everything comes with a bit of interpretation. The application and use of ontologies is very much like the application and use of written language in general. In order for written language to work well two equally important things need to happen. First, the writer needs to be able to write. They need to be able to choose the most appropriate language for their intended audience. Shakespeare is not "right" with his descriptions of love, but instead his descriptions of love (and many other human emotions) resinate with a very large number of people. Second, written language requires the reader to have a particular adeptness as well. Shakespeare can not be expected to write one thing and communicate to everybody. The reader needs to understand English, or the translation from English into another language needs to be compete and accurate.

The Internet, by design, is a decentralized environment. There are very few rules on how it is expected to be used. To a great extent it relies on sets of behavior that are more common practice as opposed to articulated rules. For example, what "rules" exist for tweets on Twitter? What rules exist for Facebook or blog postings. Creating sets of rules will not fly on the Internet because there is no over-arching governing body to enforce any rules. Sure, there are things like Dublin Core with their definitions, but

those definitions are left to interpretation, and there are no judges nor courts nor laws determining whether or not any particular application of Dublin Core is "correct". Only the common use of Dublin Core is correct, and its use is not set in stone. There are no "should's" on the Internet. There is only common practice.

With this in mind, it is best for you to work with others both inside and outside your discipline to select one or more ontologies to be used in your linked data. Do not think about this too long nor too hard. It is an never-ending process that is never correct. It is only a process that approximates the best solution.

> "The people of Rome speak Italian, mostly. But it is not difficult to hear other languages as well. Rome is an international city."

RDF serializations

> "While Romans speak Italian, mostly. There are different dialects, each with its own distinct characteristics."

RDF is a data model, and so far it has only been described (and illustrated) in the abstract. RDF needs to be exchanged between computers, and therefore it needs to be more concretely expressed. There are a number of ways RDF can be expressed, and these expressions are called "serializations".

RDF (Resource Description Framework) is a conceptual data model made up of "sentences" called triples – subjects, predicates, and objects. Subjects are expected to be URIs. Objects are expected to be URIs or string literals (think words, phrases, or numbers). Predicates are "verbs" establishing relationships between the subjects and the objects. Each triple is intended to denote a specific fact.

When the idea of the Semantic Web was first articulated XML was the predominant data structure of the time. It was seen as a way to encapsulate data that was both readable by humans as well as computers. Like any data structure, XML has both its advantages as well as disadvantages. On one hand it is easy to determine whether or not XML files are well-formed, meaning they are syntactically correct. Given a DTD, or better yet, an XML schema, it is also easy to determine whether or not an XML file is valid – meaning does it contain the necessary XML elements, attributes, and are they arranged and used in the agreed upon manner. XML also lends itself to transformations into other plain text documents through the generic, platform-independent, XSLT (Extensible Stylesheet Language Transformation) process. Consequently, RDF was originally manifested – made real and "serialized" – though the use of RDF/XML.

The example of RDF at the beginning of the Guidebook was an RDF/XML serialization:

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#" xmlns:dcterms="http://purl.org/dc/terms/"
xmlns:foaf="http://xmlns.com/foaf/0.1/">
  <rdf:Description rdf:about="http://en.wikipedia.org/wiki/Declaration_of_Independence">
    <dcterms:creator>
      <foaf:Person rdf:about="http://id.loc.gov/authorities/names/n79089957">
        <foaf:gender>male</foaf:gender>
      </foaf:Person>
    </dcterms:creator>
  </rdf:Description>
</rdf:RDF>
```

On the other hand, XML, almost by definition, is verbose. Element names are expected to be human-readable and meaningful, not obtuse nor opaque. The judicious use of special characters (&, <, >, ", and ') as well as entities only adds to the difficulty of actually reading XML. Consequently, almost from the very beginning people thought RDF/XML was not the best way to express RDF, and since then a number of other

syntaxes – serializations – have manifested themselves.

Below is the same RDF serialized in a format called Notation 3 (N3), which is very human readable, but not extraordinarily structured enough for computer processing. It incorporates the use of a line-based data structure called N-Triples used to denote the triples themselves:

```
@prefix foaf: <http://xmlns.com/foaf/0.1/>.
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix dcterms: <http://purl.org/dc/terms/>.
<http://en.wikipedia.org/wiki/Declaration_of_Independence> dcterms:creator
<http://id.loc.gov/authorities/names/n79089957>.
<http://id.loc.gov/authorities/names/n79089957> a foaf:Person;
    foaf:gender "male".
```

JSON (JavaScript Object Notation) is a popular data structure inherent to the use of JavaScript and Web browsers, and RDF can be expressed in a JSON format as well:

```
{
  "http://en.wikipedia.org/wiki/Declaration_of_Independence": {
    "http://purl.org/dc/terms/creator": [
      {
        "type": "uri",
        "value": "http://id.loc.gov/authorities/names/n79089957"
      }
    ]
  },
  "http://id.loc.gov/authorities/names/n79089957": {
    "http://xmlns.com/foaf/0.1/gender": [
      {
        "type": "literal",
        "value": "male"
      }
    ],
    "http://www.w3.org/1999/02/22-rdf-syntax-ns#type": [
      {
        "type": "uri",
        "value": "http://xmlns.com/foaf/0.1/Person"
      }
    ]
  }
}
```

Just about the newest RDF serialization is an embellishment of JSON called JSON-LD. Compare & contrasts the serialization below to the one above:

```
{
  "@graph": [
    {
      "@id": "http://en.wikipedia.org/wiki/Declaration_of_Independence",
      "http://purl.org/dc/terms/creator": {
        "@id": "http://id.loc.gov/authorities/names/n79089957"
      }
    },
    {
      "@id": "http://id.loc.gov/authorities/names/n79089957",
      "@type": "http://xmlns.com/foaf/0.1/Person",
      "http://xmlns.com/foaf/0.1/gender": "male"
```

```
    }
  ]
}
```

RDFa represents a way of expressing RDF embedded in HTML, and here is such an expression:

```
<div xmlns="http://www.w3.org/1999/xhtml"
  prefix="
    foaf: http://xmlns.com/foaf/0.1/
    rdf: http://www.w3.org/1999/02/22-rdf-syntax-ns#
    dcterms: http://purl.org/dc/terms/
    rdfs: http://www.w3.org/2000/01/rdf-schema#"
  >
  <div typeof="rdfs:Resource" about="http://en.wikipedia.org/wiki/Declaration_of_Independence">
    <div rel="dcterms:creator">
      <div typeof="foaf:Person" about="http://id.loc.gov/authorities/names/n79089957">
        <div property="foaf:gender" content="male"></div>
      </div>
    </div>
  </div>
</div>
```

The purpose of publishing linked data is to make RDF triples easily accessible. This does not necessarily mean the transformation of EAD or MARC into RDF/XML, but rather making accessible the statements of RDF within the context of the reader. In this case, the reader may be a human or some sort of computer program. Each serialization has its own strengths and weaknesses. Ideally an archive will have figure out ways exploit each of the RDF serializations for specific publishing purposes.

For a good time, play with the RDF Translator which will convert one RDF serialization into another. [8]

The RDF serialization process also highlights how data structures are moving away from a document-centric models to a statement-central models. This too has consequences for way cultural heritage institutions, like archives, think about exposing their metadata, but that is the topic of another essay.

Publishing linked data

> "Rome is a place of many neighborhoods, each with its own flavor, but in each of the neighborhoods you will experience an essence of what it means to be Roman."

Just as there are many ontologies & vocabularies, just as there are many ways to express RDF, there are many ways to publish RDF. This section introduces many of them, and they are discussed in greater detail later in the book.

Once your data has been modeled as RDF, it will be time to make it accessible on the Web. There are many ways to do this, and each of the rely on how your data/information is locally stored. Your data/information is probably already modeled in some sort of store, and it will be the job of somebody to transform that data into RDF. For example, the store might be a simple spreadsheet -- a flat file or rows & columns of information. In this model, each row in the spreadsheet will need to have a URI "minted" for it. That URI will then have predicates denoting the names of each column. Finally, the object of each RDF triple will be the value of each item in the row. The result will be set of RDF in the form of some serialization. This can be put on the Web directly, or saved in a triple store, described below.

If the data/information is modeled in a relational database, then the process of manifesting it as RDF is very similar to the process of transforming a flat file for rows & columns. There are two other options as well. One is the use of R2RML, a relational database to RDF modeling language. [9] Another option is to

use a piece of open source software called D2RQ which will convert a well-designed relational database into RDF. [10]. In either case, the resulting RDF could be saved as one or more files on a Web server or shared dynamically directly from the database.

Many people use some sort of document to model their data/information. In the world of archives, this document may be an EAD file or a set of MARC records. While not ideal, it is possible to use XSL to transform the EAD or MARC into a RDF serializations. This is the process employed by the Guidebook's "sandbox" application. This is the same process used by a number of more formal projects including Linking Lives and ReLoad. [5, 11].

If RDF serializations are saved as files on the Web, as in the examples outlined above, then those files may be simple dumps of data which are large files consisting of huge numbers of triples. The URLs of those files are then listed in some sort of Web page intended for people to read. Sometimes the locations of these RDF files are listed in a different type of file called a VOID file. [12]

Alternatively, RDF might be published side-by-side their human readable HTML counter-parts. The VIAF URIs are good examples. If this method is employed, then a process called "content negotiation" is expected to be implemented. Content negotiation is an integral part of the way the World Wide Web operates. It is method for one computer to request a specific form of data/information from a server. In terms of linked data, a computer can request the content of a URI in the form of HTML or RDF, and the remote server will provide the information in the desired format, if available. Content negotiation is described in greater detail later in the Guidebook.

Some people's data/information may be modeled in a content management system, like Drupal. These systems generate HTML on-the-fly. In these cases it is a good idea embed RDF into the HTML using RDFa. [13] RDFa is also an option when generating HTML out of databases on-the-fly.

Finally, there exist "databases" specifically designed to store RDF triples. These "databases" are called "triple stores". Along side these triple stores are methods for searching their contents. This searching mechanism is called SPARQL, and SPARQL "endpoints" may be available on the Web. A number of triple stores are lists later in the Guidebook, and a SPARQL tutorial is available as well.

In summary, there are many ways of publishing linked data. At first glance, it may seem as if there are too many choice and difficult to implement, but in reality modeling your data as RDF is much more challenging.

> "Rome has many places to eat. Food is available in cafes, open air markets, family owned grocery stores, and commercial conglomerates. There are may ways to get food to the consumer."

Linked open data

> "Rome has long understood the benefits of common areas, hence freely accessible squares, fountains, and market places."

Some people make a distinction between linked data and linked open data (LOD). Linked data is what has been described so far in this guidebook. It is a process for making data and metadata available on the Web using RDF as the underlying data model and incorporating into it as many links (URIs) as possible. This process does not necessarily stipulate any intellectual property claims on the published data and metadata. (Intellectual property claims can be explicitly stated through the use of VOID files -- sets of triples using the VOID ontology and published along side RDF files.) For the most part data and metadata accessible via linked data is assumed to free, as in gratis and without financial obligation. At the same time, consumers of linked data are expected to acknowledge the time and effort others have spent in the creation of the consumed data, and consumers are not expected to call the data their own. This is an unspoken courtesy.

LOD is linked data that is explicitly denoted as free, as in gratis and without financial obligation. The idea seems to have been born from the idea of all things open (open source software and open access publishing). There is a strong community of LOD advocates in libraries, archives, and museums. The community is called LODLAM and has sponsored a few summits. [14] The Open Knowledge Foundation is also a ver strong advocate for LOD. [15]

Strictly speaking linked data is a process, and linked open data is a thing. For all intents and purposes, this guidebook assumes the data and information made accessible via linked data is linked open data, but there might come a time in the future when access controls are places against some linked data. The data and information of linked data is not necessarily provided gratis.

Consuming linked data

> "Rome, like any other city, is full of give and take."

Publishing linked data is only half of the equation. Consuming linked data is the other half. Without considering the consuming part of linked data, it is not possible to reap all of the benefits linked data has to offer.

No primer on linked data would be complete without the inclusion of the famous LOD cloud diagram, below:

> [INSERT LOD CLOUD DIAGRAM HERE.]

While the diagram has not been updated in a few years, it represents the sets of published linked data available. It also illustrates how they relate to each other. As you can see DBedia is at the center of the cloud and illustrates how other data sets rely on it for (some) content. Many, if not all, of the sets of RDF have been registered in a directory called Datahub. [16] There one can search and browse for all sorts of data sets access via linked data standards as well as other protocols. Datahub is a good place to look for complementary data sets to your own.

It is not possible for anybody to completely describe any collection of data. All data and information is inextricably linked to other data and information. There is not enough time nor energy in the world for any individual nor discrete group to do all the work. By harnessing the collective power of the Web (and linked data), it is possible to create more complete collections and more thorough description. These more complete collections and more complete descriptions can be created in two different ways. The first has already been described -- through the judicious use of shared URIs. The second is by harvesting linked data from other locations, combining it with your own, and producing value-added services against the result.

About linked data, a review

Implementing linked data represents a different, more modern way of accomplishing some of the same goals of archival science. It is a process of making more people aware of your content. It is not the only way to make more people aware, but it represents a way that will be wide spread, thorough, and complete.

Linked data, or more recently referred to as "linked open data" for reasons to be explained later, is a proposed technique for generating new knowledge. It is intended to be a synergy between people and sets of agreed upon computer systems that when combined will enable both people and computers to discover and build relationships between seemingly disparate data and information to create and discover new knowledge.

In a nutshell, this is how it works. People possess data and information. They encode that data and information in any number of formats easily readable by computers. They then make the encoded data and information available on the Web. Computers are then employed to systematically harvested the encoded data. Since the data is easily readable, the computers store the data locally and look for similarly encoded things in other locally stored data sets. When similar items are identified relationships can be

inferred between the items as well as the other items in the data set. To people, some of these relationships may seem obvious and "old hat". On the other hand, since the data sets can be massive, relationships that were never observed previously may come to light, thus new knowledge is created.

Some of this knowledge may be trivial. For example, there might be a data set of places -- places from all over the world including things like geographic coordinates, histories of the places, images, etc. There might be another data set of poeple. Each person may be described using their name, their place of birth, and a short biography. These data sets may contain ten's of thousands of items each. Using linked data it would be possible to cross reference the people with the places to discover who might have met whom when and where. Some people may have similar ideas, and those ideas may have been generated in a particular place. Linked data may help in discovering who was in the same place at the same time and the researcher may be better able to figure out how a particular idea came to fruition.

Here's an example hitting closer to the home of archives and archivists. Suppose most archival finding aids were written in a format easily readable by computers. Let's call this format Encoded Archival Description. Let's suppose these finding aids were made available on the Web. Let's suppose one or more computers crawled these archival sites harvesting the finding aids. Once done a computer program could be used to find all the occurrences of particular name and generate a virtual finding aid that is more complete and more comprehensible than any single finding aid on that particular person.

The amount of data and information accessible today is greater in size than it has ever been in human history. Using our traditional techniques of reading, re-reading, writing, discussing, etc. is more than possible to learn new things about the state of the world, the universe, and the human condition. By exploiting the current state of computer technology is possible to expand upon our traditional techniques and possibly accelerate the mass of knowledge.

When you hear of linked data and the Semantic Web, the next thing you often hear is "RDF" or "Resource Description Framework". First and foremost, RDF is a way of representing knowledge. It does this through the use of assertions (think, "sentences") with only three parts: 1) a subject, 2) a predicate, and 3) an object. Put together, these three things create things called "triples". T

RDF is not to be confused with RDF/XML or any other type of RDF "serialization". Remember, RDF describes triples, but it does not specify how the triples are express or written down. On the other hand, RDF/XML is an XML syntax for expressing RDF. Some people think RDF/XML is too complicated and too verbose. Consequently, other serializations have manifested themselves including N3 and Turtle.

In "Linked Data -- Design Issues" Berners-Lee outlined four often-quoted expectations for implementing the Semantic Web. Each of these expectations are listed below along with some elaborations:

* "Use URIs as names for things" - URIs (Universal Resource Identifiers) are unique identifiers, and they are expected to have the same shape as URLs (Universal Resource Locators). These identifiers are expected to represent things such as people, places, institutions, concepts, books, etc. URIs are monikers or handles for real world or imaginary objects.

* "Use HTTP URIs so that people can look up those names." - The URIs are expected to look and ideally function on the World Wide Web through the Hypertext Transfer Protocol (HTTP), meaning the URI's point to things on Web servers.

* "When someone looks up a URI, provide useful information, using the standards (RDF, SPARQL)" - When URIs are sent to Web servers by Web browsers (or "user-agents" in HTTP parlance), the response from the server should be in a conventional, computer readable format. This format is usually a "serialization" of RDF (Resource Description Framework) -- a notation looking much like a

rudimentary sentence composed of a subject, predicate, and object.

* "Include links to other URIs. So that they can discover more things." - Simply put, try very hard to use URIs other people have have used. This way the relationships you create can literally be linked to the relationships other people have created. These links may represent new knowledge.

In the same text Berners-Lee also outlined a sort of reward system -- a sets of stars -- for levels of implementation. This reward system also works very well as a strategy for publishing linked data by cultural heritage institutions such as archives. A person gets:

* one star for making data available on the web (in whatever format) but with an open license

* two stars for making the data machine-readable and structured data (e.g. Excel instead of an image scan of a table)

* three stars for making the data available in a non-proprietary format (e.g. comma-separated values instead of Excel)

* four stars for using open standards from W3C (RDF and SPARQL) to identify things, so that people can point at your stuff

* five stars for linking your data to other people's data to provide context

Linked data, today

4.b. Trends in LOD-LAM

* Mash ups
* Harvesting along side other protocols
* Increased interest
* Increased number of RDF serializations
* Governments making their content available
* Using them to enhance online catalogs
* Creating timelines
* Creating "named graphs"
* Increased number of programming toolkits
* Emphasis on "open" linked data and linked data in museums and archives
* Making RDF dumps available
* Interest in schema.org

With great interest I read the Spring/Summer issue of Information Standards Quarterly where there were a number of articles pertaining to linked open data in cultural heritage institutions. [0] Of particular interest to me where the various loosely enumerated challenges of linked open data. Some of them included:

* the apparent Tower Of Babel when it comes to vocabularies used to describe content, and the same time we need to have "ontology mindfulness".
* dirty, inconsistent, or wide varieties of data integrity
* persistent URIs
* the "chicken & egg" problem of why linked data if there is no killer application

There are a number of challenges in the process. Some of them are listed below, and some of them have been alluded to above:

* Create useful LOD, meaning, create LOD that links to other LOD. LOD does not live in a world by itself. Remember, the "L" stands for "linked". For example, try to include URIs that are the URIs used on other LOD data sets. Sometimes this is not possible, for example, le with the names of people in archival materials. When possible, they used VIAF, but other times they needed to create their own URI denoting an individual.

* There is a level of rigor involved in creating the data model, and there may be many discussions regarding semantics. For example, what is a creator? Or, when is a term intended to be an index term as opposed reference. When does one term in one vocabulary equal a different term in a different vocabulary?

* Balance the creation of your own vocabulary with the need to speak the language of others using their vocabulary.

* Consider "fixing" the data as it comes in or goes out because it might not be consistent nor thorough.

* Provenance is an issue. People – especially scholars – will want to know where the LOD came from and whether or not it is authoritative. How to solve or address this problem? The jury is still out on this one.

* Creating and maintaining LOD is difficult because it requires the skills of a number of different types of people. Computer programmers. Database designers. Subject experts. Metadata specialists. Archivists. Etc. A team is all but necessary.

Benefits

Archives are about collecting, organizing, preserving, and disseminating original, unique, and primary literature. These are the whats of archival practice, but the hows of archival practice evolve with the changing technology. With the advent of ubiquitous networked computing, people's expectations regarding access to information and knowledge have changed significantly. Unless institutions like archives change with the times, then the needs previously filled by archives will be filled by other institutions. Linked data is a how of archival practice, and it is one of those changes behooving archives to adopt. It is a standards-based technique for making data and information available on the Web. It is rooted in the very fabric of the Web and therefore is not beholden to any particular constituency. It is a long lasting standard and practice that will last as long as the hypertext transfer protocol is operational.

Making archival descriptions and collection available via linked data will increase the use of those descriptions and collections. It is a form of benign advertising. Commercial search engines will harvest the linked data content and make it available it their search engines. Search engines will return hits to your descriptions and collections driving traffic to you and your site. Digital humanists will harvest your content, perform analysis against it, and create new knowledge or bring hidden knowledge to light. Computer scientist will collect your data, amalgamate it with the data of others, and discover relationship previously unconceived.

You can divide your combined collections and services into two tangible parts: 1) the collections themselves, and 2) the metadata describing them. It is usually possible to digitize your collections, but the result is rarely 100% satisfactory. Digitization is almost always a useful surrogate not a complete replacement. In this way, your collections as physical objects will always be a draw to all types of learners and researchers. The metadata, on the other hand, is 100% digitizable, and therefore lends itself very well to dissemination on the Internet. Linked data represents one way to make this happen.

Few archival collections are 100% complete. There are always pieces missing, and some of those missing pieced will be owned by others. Your collections will have relationship with other collection, but you

will not have direct access to those other collections. Some of these relationships are explicit. Some of them are implicit. If everybody were to expose their metadata then those explicit and implicit relationships can become more apparent. Once these relationships are strengthened and become more obvious, interest in the collections will increase accordingly, and the collections will be used to a greater degree. With this increased use will come increased attention, and in turn, a greater measure of success for the collections and services it provides.

From: Ingrid Mason <ingrid.b.mason@gmail.com>
Subject: Re: [LODLAM] quick benefits to hosting instutitions
Date: January 22, 2014 at 9:49:47 PM EST
To: lod-lam@googlegroups.com
Reply-To: lod-lam@googlegroups.com

Hi Jody,

If I understand correctly, you're keen to find examples of value generated. I'll give this a whirl... and see if I'm being helpful.

Converting data that already exists, >> providing a data service that a user community seeks to reuse and contribute to (access value).

PeopleAustralia (National Library of Australia) provides permanent, resolvable unique identifiers that link to records about parties, i.e. people or organisations. The authority file at the Library already had, was reused. This data source was enhanced as a result of ANDS funding to identify parties that manage or own research data collections. You'll see that this has increased the capacity for discovery (through collaboration with other data providers). Key contact: Tim Sherratt @wragge

Collaborating with custodians of primary material (collection managers), using third party data (Dbpedia), and finding, identifying and linking entities in the data >> brought to light information that was previously unknown (research value).

LinkedJazz at the Pratt Institute. Finalist in the LODLAM 2013 summit award along with some other folks (for being generally super clever with LOD things). Key contact: Cristina Patuelli @cristinapattuel

Providing insights into the links in your own data >> improve data quality (data value).

Check out Chris McDowall's post on linking data in digitalNZ. @fogonwater

There are other kinds of value in all that. Guess benefits depend on what the strategic goals of the organisation are, and what the research community needs in terms of access.

Hope that helps?

Ingrid

> "Rome is a large city that keeps getting larger. It is built on
> rich traditions, and the city continues to evolve. The language
> of the Romans expressive, and people speak with more than just
> words. They also speak with their hands. There are many ways to
> enjoy Rome. Different areas will appeal to different people, but
> to really understand Rome as a whole, a person need to visit and
> appreciate each of them in turn."

Links

[1] canonical article - http://csis.pace.edu/~marchese/CS835/Lec9/112_SemWeb.pdf
[2] design issues - http://www.w3.org/DesignIssues/LinkedData.html
[3] FOAF - http://www.foaf-project.org
[4] DCMI Metadata Terms - http://dublincore.org/documents/dcmi-terms/
[5] Linking Lives project - http://archiveshub.ac.uk/linkinglives/
[6] Linking Lives ontology - http://data.archiveshub.ac.uk/def/
[7] VIAF - http://viaf.org
[8] RDF Translator - http://rdf-translator.appspot.com
[9] R2RML - http://www.w3.org/TR/r2rml/
[10] D2RQ - http://d2rq.org/
[11] ReLoad - http://labs.regesta.com/progettoReload/en
[12] VOID - http://www.w3.org/TR/void/
[13] RDFa - http://www.w3.org/TR/xhtml-rdfa-primer/
[14] LODLAM - http://lodlam.net
[15] Open Knowledge Foundation - http://okfn.org
[16] Datahub - http://datahub.io

5. Getting Started: Strategies and Steps

Linked data represents a modern way of making your archival descriptions accessible to the wider world. In that light, it represents a different way of doing things but not necessary a different what of doing things. You will still be doing inventory. You will still be curating collections. You will still be prioritizing what goes and what stays.

On the other hand, linked data changes the way your descriptions get expressed and distributed. It is a lot like taking a trip across country. The goal was always to get to the coast to see the ocean, but instead of walking, going by stage coach, taking a train, or driving a car, you will be flying. Along the way you may visit a few cities and have a few layovers. Bad weather may even get in the way, but sooner or later you will get to your destination. Take a deep breath. Understand that the process will be one of learning, and that learning will be applicable in other aspects of your work. The result will be two-fold. First, a greater number of people will have access to your collections, and consequently, more people will will be using your collections.

5.a. Defining your strategy: Articulate goals, objectives, and metrics to measure success.

The building blocks of linked data include:

  * URIs pointing to real-world objects: people, places, or things where things can be ideas or just about anything on the Web
  * Ontologies, the language(s) of relationships between the URIs
  * Content to share with the wider world
  * People to do the work
  * Computer technology to manifest the work

With this in mind, articulate some goals – broad targets of things you would like to accomplish. Some of them might include:

  * making your archival collections more widely accessible
  * working with others to build virtual collections of like topics or formats
  * incorporating your archival descriptions into public spaces like Wikipedia
  * integrating your collections into local teaching, learning, and research activities
  * increasing the awareness of your archive to benefactors
  * increasing the computer technology skills of fellow archivists

How might you go about accomplishing these goals? What are your objectives? (What method of transportation are you going to use to get where you are going?) How am I going to measure success? In other words, you will need to create an plan, and each item in the plan answers a simple question – Who is going to do what by when? In other word, what people will be responsible for accomplishing the particular objective. Exactly what will they be doing, and by what time will they have it accomplished. Each of these components are described in greater detail below

Who

It is quite unlikely your linked data goals and objectives will be accomplished by a single person. Instead it will most likely required a team of people. These people do not necessarily need to working in the same physical location, but they will require a diverse set of skills. Some of them include, and each plays a key, indispensable role:

  * content specialists - These are the people who understand the “aboutness” of a particular collection. These are the people who understand and can thoroughly articulate the significance of a collection. They know how and why particular things belong in a collection. They are able to answer questions about the collection as all as tell stories against it.

  * metadata specialists - These are people who understand data about data. Not only do they understand the principles of controlled vocabularies and authority lists, but they are also familiar with a wide variety of such lists, specifically as they are represented on the Web. In linked data there are fewer descriptive cataloging “rules”. Nevertheless, the way the ontologies of linked data can be used need to be interpreted, and this interpretation needs to be consistent. Metadata specialists understand these principles.

  * computer technologists - Not only are these the people who have a fundamental understanding of what computer can and cannot do, but they also know how to put this understanding into practice. At the very least, the computer technologists need to understand a myriad of data structures and how to convert them into different data structures. Converting MARC 21 into MARCXML. Transforming EAD into HTML. Reporting against a relational database to create serialized RDF. These tasks required computer programming skills, but not necessarily any one in particular. Any modern programming language (Java, PHP, Python, Ruby, etc.) includes the necessary function to complete the tasks.

What

The what of your objectives are not so much identified with nouns as they are action verbs, such as: write, evaluate, implement, examine, purchase, hire, prioritize, list, delete, acquire, discuss, share, find, compare & contrast, stop, start, complete, continue, describe, edit, updated, create, purchase, upgrade, etc. The what of your objective is in the doing.

When

The say, “Work expands to fill the available space.” If this is true, and no deadlines are articulated for each objective, then the allotted amount of time for any given task is all but infinite, but this it not true. Time is one of the most limited resources you have. When thinking about a given objective, ask yourself how much time you think it will take, multiply the time by one and a half. Ask yourself when the task can begin and document the beginning point as well as the estimated ending point. Do this all of your objectives and the result will be a Gantt chart. It will now be easy to look at the chart on a regular basis to see who things are progressing.

---

1. Create and maintain RDF - Generate, save, and update sets of RDF statements describing your content.

2. Publish RDF - No matter what kind of RDF you are able to create, make it available on the Web.

3. Articulate and implement best practices for publishing RDF - Work with your friends to articulate and document an "application profile". As guidelines and best practices get articulated, implement them by going back to Step #1. In the meantime, continue on to Step #4.

4. Harvest RDF - Collect RDF from the Web, and some of the collection may be some of your own content.

5. Develop services against harvested RDF - Evaluate (curate) the collection.

6. Go to Step #1 - This is a never-ending process.

--

2.a Lots of ways to participate

  * finding URIs
  * associating URIs with predicates
  * publishing serialized RDF with: 1) pure HTTP, and/or 2) SPARQL
  * harvesting RDF
  * storing RDF
  * analyzing RDF
  * providing services against RDF
  * leading groups of people
  * articulating policies
  * allocating resources

3.a. Objectives: management, access, and use and linked data affordances

[Management, access, and use and linked data affordances]

5.a Linked data and archival practice: Or, There is more than one way to get there

Two recent experiences have taught me that – when creating some sort of information service – linked data will reside and be mixed in with data collected from any number of Internet techniques. Linked data interfaces will coexist with REST-ful interfaces, or even things as rudimentary as FTP. To the archivist, this means linked data is not the be-all and end-all of information publishing. There is no such thing. To the application programmer, this means you will need to have experience with a ever-growing number of Internet protocols. To both it means, "There is more than one way to get there."

In October of 2013 I had the opportunity to attend the Semantic Web In Libraries conference. It was a three-day event attended by approximately three hundred people who could roughly be divided into two equally sized groups: computer scientists and cultural heritage institution employees. The bulk of the presentations fell into two categories: 1) publishing linked data, and 2) creating information services. The publishers talked about ontologies, human-computer interfaces for data creation/maintenance, and systems exposing RDF to the wider world. The people creating information services were invariably collecting, homogenizing, and adding value to data gathered from a diverse set of information services. These information services were not limited to sets of linked data. They also included services accessible via REST-ful computing techniques, OAI-PMH interfaces, and there were probably a few locally developed file transfers or relational database dumps described as well. These people where creating lists of information services, regularly harvesting content from the services, writing cross-walks, locally storing the content, indexing it, providing services against the result, and sometimes republishing any number of "stories" based on the data. For the second group of people, linked data was certainly not the only game in town.

In February of 2014 I had the opportunity to attend a hackathon called GLAM Hack Philly. A wide variety of data sets were presented for "hacking" against. Some where TEI files describing Icelandic manuscripts. Some was linked data published from the British museum. Some was XML describing digitized journals created by a vendor-based application. Some of it resided in proprietary database applications describing the location of houses in Philadelphia. Some of it had little or no computer-readable structure at all and described plants. Some of it was the wiki mark-up for local municipalities. After the attendees (there were about two dozen of us) learned about each of the data sets we self-selected and hacked away at projects of our own design. The results fell into roughly three categories: geo-referencing objects, creating searchable/browsable interfaces, and data enhancement. With the exception of the resulting hack repurposing journal content to create new art, the results were pretty typical for cultural heritage institutions. But what fascinated me was way us hackers selected our data sets. Namely, the more complete and well-structured the data was the more hackers gravitated towards it. Of all the data sets, the TEI files were the most complete, accurate, and computer-readable. Three or four projects were done against the TEI. (Heck, I even hacked on the TEI files.) The linked data from the British Museum – very well structured but not quite as through at the TEI – attracted a large number of hackers who worked together for a common goal. All the other data sets had only one or two people working on them. What is the moral to the story? There are two of them. First, archivists, if you want people to process your data and do "kewl" things against it, then make sure the data is thorough, complete, and computer-readable. Second, computer programmers, you will need to know a variety of data formats. Linked data is not the only game in town.

In summary, the technologies described in this Guidebook are not the only way to accomplish the goals of archivists wishing to make their content more accessible. Instead, linked data is just one of many protocols in the toolbox. It is open, standards-based, and simpler rather than more complex. On the other hand, other protocols exist which have a different set of strengths and weaknesses. Computer technologists will need to have a larger rather than smaller knowledge of various Internet tools. For archivists, the core of the problem is still the collection and description of content. This – a what of archival practice – continues to remain constant. It is the how of archival practice – the technology – that changes at a much faster pace.

2.a Use cases

What can you do with linked data once it is created? Here are three use cases:

1. Do simple publishing - At its very root, linked data is about making your data available for others to harvest and use. While the “killer linked data application” has seemingly not reared its head, this does not mean you ought not make your data available at linked data. You won’t see the benefits immediately, but sooner or later (less than 5 years from now), you will see your content creeping into the search results of Internet indexes, into the work of both computational humanists and scientists, and into the hands of esoteric hackers creating one-off applications. Internet search engines will create “knowledge graphs”, and they will include links to your content. The humanists and scientists will operate on your data similarly. Both will create visualizations illustrating trends. They will both quantifiably analyze your content looking for patterns and anomalies. Both will probably create network diagrams demonstrating the flow and interconnection of knowledge and ideas through time and space. The humanist might do all this in order to bring history to life or demonstrate how one writer influenced another. The scientist might study ways to efficiently store your data, easily move it around the Internet, or connect it with data set created by their apparatus. The hacker (those are the good guys) will create flashy-looking applications that many will think are weird and useless, but the applications will demonstrate how the technology can be exploited. These applications will inspire others, be here one day and gone the next, and over time, become more useful and sophisticated.

2. Create a union catalog - If you make your data available as linked data, and if you find at least one other archive who is making their data available as linked data, then you can find a third somebody who will combine them into a triple store and implement a rudimentary SPARQL interface against the union. Once this is done a researcher could conceivably search the interface for a URI to see what is in both collections. The absolute imperative key to success for this to work is the judicious inclusion of URIs in both data sets. This scenario becomes even more enticing with the inclusion of two additional things. First, the more collections in the triple store the better. You can not have enough collections in the store. Second, the scenario will be even more enticing when each archive publishes their data using similar ontologies as everybody else. Success does not hinge on similar ontologies, but success is significantly enhanced. Just like the relational databases of today, nobody will be expected to query them using their native query language (SQL or SPARQL). Instead the interfaces will be much more user-friendly. The properties of classes in ontologies will become facets for searching and browsing. Free text as well as fielded searching via drop-down menus will become available. As time goes on and things mature, the output from these interfaces will be increasingly informative, easy-to-read, and computable. This means the output will answer questions, be visually appealing, as well as be available in one or more formats for other computer programs to operate upon.

3. Tell a story - You and your hosting institution(s) have something significant to offer. It is not just about you and your archive but also about libraries, museums, the local municipality, etc. As a whole you are a local geographic entity. You represent something significant with a story to tell. Combine your linked data with the linked data of others in your immediate area. The ontologies will be a total hodgepodge, at least at first. Now provide a search engine against the result. Maybe you begin with local libraries or museums. If you work in an academic setting, then maybe you begin with other academic departments across campus. Allow people to search the interface and bring together the content of everybody involved. Do not just provide lists of links in search results, but instead create knowledge graphs. Supplement the output of search results with the linked data from Wikipedia, Flickr, etc. In a federated search sort of way, supplement the output with content from other data feeds such as (licensed) bibliographic indexes or content harvested from OAI-PMH repositories. Identify complementary content from further afield. Figure out a way for you and they to work together to create a newer, more complete set of content. Creating these sorts of things on-the-fly will be challenging. On the other hand, you might implement something that is more iterative and less immediate, but more thorough and curated if you were to select a topic or theme of interest, and do your own searching and story telling. The result would be something that is at once a Web page, a document designed for printing, or something importable into another computer program.

4. Create new knowledge - Create an inference engine, turn it against your triple store, and look for

relationships between distinct sets of URIs that weren't previously apparent. Here's one way how:

1. allow the reader to select an actionable URI of personal interest, ideally a URI from the set of URIs you curate

2. submit it an HTTP server or SPARQL endpoint and request RDF as output

3. save the output to a local store

4. for each subject and object URI found the output, go to Step #2

5. go to step #2 n times for each newly harvested URI in the store where n is a reader-defined integer greater than 1; in other words, harvest more and more URIs, predicates, and literals based on the previously harvested URIs

6. create a set of human readable services/reports against the content of the store, and think of these services/reports akin to finding aids, reference materials, or museum exhibits of the future: Example services/reports might include:

    * hierarchal lists of all classes and properties - This would be a sort of semantic map. Each item on the map would be clickable allowing the reader to read more and drill down.

    * text mining reports - collect into a single "bag of words" all the literals saved in the store and create: word clouds, alphabetical lists, concordances, bibliographies, directories, gazetteers, tabulations of parts of speech, named entities, sentiment analyses, topic models, etc.

    * maps - use place names and geographic coordinates to implement a geographic information service

    * audio-visual mash-ups - bring together all the media information and create things like slideshows, movies, analyses of colors, shapes, patterns, etc.

    * search interfaces - implement a search interface against the result, SPARQL or otherwise

    * facts - remember SPARQL queries can return more than just lists. They can return mathematical results such as sums, ratios, standard deviations, etc. It can also return Boolean values helpful in answering yes/no questions. You could have a set of canned fact queries such as, how many ontologies are represented in the store. Is the number of ontologies greater than 3? Are there more than 100 names represented in this set? The count of languages used in the set, etc.

7. Allow the reader to identify a new URI of personal interest, specifically one garnered from the reports generated in Step #5.

8. Go to Step #2, but this time have the inference engine be more selective by having it try to crawl back to your namespace and set of locally curated URIs.

9. Return to the reader the URIs identified in Step #7, and by consequence, these URIs ought to share some of the same characteristics as the very first URI; you have implemented a "find more like this one" tool. You, as curator of the collection of URIs might have thought the relations between the first URI and set of final URIs was obvious, but those relationships would not necessarily be obvious to the reader, and therefore new knowledge would have been created or brought to light.

10. If there are no new URIs from Step #7, then go to Step #6 using the newly harvested content.

11. Done - if a system were created such as the one above, then the reader would quite likely have acquired some new knowledge, and this would be especially true the greater the size of n in Step #5.

5.b. Is your archival description LOD-ready?

Is your archival description LOD-ready? Now? The simple, straight-forward answer is, "Yes." The longer and more complicated answer is, "No. Your data is never 100% linked data ready because the process of archival description is never finished." That said, the balance of the Guide describes what you can do going forward.

5.c. Identify building blocks: metadata components in archival description that are (or nearly are) ready for linking.

5.d. Readiness: Making small changes in practice to make your description LOD-ready.

5.e. What you can do now if you have (done)

Each of the sections below outline how you can participate in linked data if currently have any number of metadata file formats (MARC, EAD, etc.).

Please remember, RDF is really about sets of triples, and these sets do not neatly correspond to document structures like MARC or EAD. Using MARC and/or EAD to publish linked data is functional but not necessarily optimal. At the same time, there does not currently exist a method for people writing archival description to directly publish the fruits of their labors as RDF. Until such methods present themselves, transforming present dauy metadata file formats is a viable option.

5.e.i. EAD (done)

If you have used EAD to describe your collections, then you can easily make your descriptions available as valid linked data, but the result will be less than optimal. This is true not for a lack of technology but rather from the inherent purpose and structure of EAD files.

A few years ago an organisation in the United Kingdom called the Archive's Hub was funded by a granting agency called JISC to explore the publishing of archival descriptions as linked data. One of the outcomes of this effort was the creation of an XSL stylesheet transforming EAD into RDF/XML. The terms used in the stylesheet originate from quite a number of standardized, widely accepted ontologies, and with only the tiniest bit configuration / customization the stylesheet can transform a generic EAD file into valid RDF/XML. The resulting XML files can then be made available on a Web server or incorporated into a triple store. This goes a long way to publishing archival descriptions as linked data. The only additional things needed are a transformation of EAD into HTML and the configuration of a Web server to do content negotiation between the XML and HTML.

For the smaller archive with only a few hundred EAD files whose content does not change very quickly, this is a simple, feasible, and practical solution to publishing archival descriptions as linked data. With the exception of doing some content negotiation, this solution does not require any computer technology that is not already being used in archives, and it only requires a few small tweaks to a given workflow:

1. implement a content negotiation solution
2. create and maintain EAD file
3. transform EAD into RDF/XML
4. transform EAD into HTML
5. save the resulting XML and HTML files on a Web server
6. go to step #2

EAD is a combination of narrative description and a hierarchal inventory list, and this data structure does not lend itself very well to the triples of linked data. For example, EAD headers are full of controlled vocabularies terms but there is no way to link these terms with specific inventory items. This is because the vocabulary terms are expected to describe the collection as a whole, not individual things. This problem could be overcome if each individual component of the EAD were associated with controlled vocabulary terms, but this would significantly increase the amount of work needed to create the EAD files in the first place.

The common practice of using literals ("strings") to denote the names of people, places, and things in EAD files would also need to be changed in order to fully realize the vision of linked data. Specifically, it would be necessary for archivists to supplement their EAD files with commonly used URIs denoting subject headings and named authorities. These URIs could be inserted into id attributes throughout an EAD file, and the resulting RDF would be more linkable, but the labor to do so would increase, especially since many of the named items will not exist in standardized authority lists.

Despite these short comings, transforming EAD files into some sort of serialized RDF goes a long way towards publishing archival descriptions as linked data. This particular process is a good beginning and

outputs valid information, just information that is not as linkable as possible. This process lends itself to iterative improvements, and outputting something is better than outputting nothing. But this particular proces is not for everybody. The archive whose content changes quickly, the archive with copious numbers of collections, or the archive wishing to publish the most compliant linked data possible will probably not want to use EAD files as the root of their publishing system. Instead some sort of database application is probably the best solution.

5.e.ii. EAC-CPF

Encoded Archival Context for Corporate Bodies, Persons, and Families (EAC-CPF) goes a long way to implementing a named authority database that could be linked from archival descriptions. These XML files could easily be transformed into serialized RDF and therefore linked data. The resulting URIs could then be incorporated into archival descriptions making the descriptions richer and more complete.

For example the FindAndConnect site in Australia uses EAC-CPF under the hood to disseminate information about people in its collection. [1] Similarly, “SNAC aims to not only make the [EAC-CPF] records more easily discovered and accessed but also, and at the same time, build an unprecedented resource that provides access to the socio-historical contexts (which includes people, families, and corporate bodies) in which the records were created” -- u More than a thousand EAC-CPF records are available from the RAMP project -- http://demo.rampeditor.info/export.php

  * Library, The standard EAC-CPF is maintained by the Society of American Archivists in partnership with the Berlin State. “Society of American Archivists and the Berlin State Library (http://eac.staatsbibliothek-berlin.de/) -

[1] FindAndConnect - http://www.findandconnect.gov.au
[2] SNAC - http://socialarchive.iath.virginia.edu

5.e.iii. MARC (done)

In some ways MARC lends it self very well to being published via linked data, but in the long run it is not really a feasible data structure.

Converting MARC into serialized RDF through XSLT is at least a two step process. The first step is to convert MARC into MARCXML. This can be done with any number of scripting languages and toolboxes. The second step is to use a stylesheet such as the one created by Stefano Mazzocchi to transform the MARCXML into RDF/XML. [1] From there a person could save the resulting XML files on a Web server, enhance access via content negotiation, and called it linked data.

Unfortunately, this particular approach has a number of drawbacks. First and foremost, the MARC format had no place to denote URIs; MARC records are made up almost entirely of literals. Sure, URIs can be constructed from various control numbers, but things like authors, titles, subject headings, and added entries will most certainly be strings ("Mark Twain", "Adventures of Huckleberry Finn", "Bildungsroman", or "Samuel Clemans"), not URIs. This issue can be overcome if the MARCXML were first converted into MODS and URIs were inserted into id or xlink attributes of bibliographic elements, but this is extra work. If an archive were to take this approach, then it would also behoove them to use MODS as their data structure of choice, not MARC. Continually converting from MARC to MARCXML to MODS would be expensive in terms of time. Moreover, with each new conversion the URIs from previous iterations would need to be re-created.

[1] stylesheet by Mazzocchi - https://github.com/dltj/MARC-MODS-RDFizer/blob/master/stylesheets/mods2rdf.xslt

5.e.iv. METS, MODS, and perhaps more. (done)

If you have archival descriptions in either of the METS or MODS formats, then transforming them into RDF is as far away as your XSLT processor and a content negotiation implementation. As of this writing there do not seem to be any METS to RDF stylesheets, but there are a couple stylesheets for MODS. The biggest issue with these sorts of implementations are the URIs. It will be necessary for archivists to include URIs into as many MODS id or xlink attributes as possible. The same thing holds true for METS files except the id attribute is not designed to hold pointers to external sites.

5.e.v. Databases (done)

Publishing linked data through XML transformation is functional but not optimal. Publishing linked data from a database comes closer to the ideal but requires a greater amount of technical computer infrastructure and expertise.

Databases -- specifically, relational databases -- are the current best practice for organizing data. As you may or may not know, relational databases are made up of many tables of data joined together with keys. For example, a book may be assigned a unique identifier. The book has many characteristics such as a title, number of pages, size, descriptive note, etc. Some of the characteristics are shared by other books, like authors and subjects. In a relational database these shared characteristics would be saved in additional tables, and they would be joined to a specific book through the use of unique identifiers (keys). Given this sort of data structure, reports can be created from the database describing its content. Similarly, queries can be applied against the database to uncover relationships that may not be apparent at first glance or buried in reports. The power of relational databases lies in the use of keys to make relationships between rows in one table and rows in other tables.

Not coincidently, this is very much the way linked data is expected to be implemented. In the linked data world, the subjects of triples are URIs (think database keys). Each URI is associated with one or more predicates (think the characteristics in the book example). Each triple then has an object, and these objects take the form of literals or other URIs. In the book example, the object could be “Adventures Of Huckleberry Finn” or a URI pointing to Mark Twain. The reports of relational databases are analogous to RDF serializations, and SQL (the relational database query language) is analogous to SPARQL, the query language of RDF triple stores. Because of the close similarity between well-designed relational databases and linked data principles, the publishing of linked data directly from relational databases makes whole lot of sense, but the process requires the combined time and skills of a number of different people: content specialists, database designers, and computer programmers. Consequently, the process of publishing linked data from relational databases may be optimal, but it is more expensive.

Thankfully, most archivists probably use some sort of database to manage their collections and create their finding aids. Moreover, archivists probably use one of three or four tools for this purpose: Archivist’s Toolkit, Archon, ArchivesSpace, or PastPerfect. Each of these systems have a relational database at their heart. Reports could be written against the underlying databases to generate serialized RDF and thus begin the process of publishing linked data. Doing this from scratch would be difficult, as well as inefficient because many people would be starting out with the same database structure but creating a multitude of varying outputs. Consequently, there are two alternatives. The first is to use a generic database application to RDF publishing platform called D2RQ. The second is for the community to join together and create a holistic RDF publishing system based on the database(s) used in archives.

D2RQ is a very powerful software system. [1] It is supported, well-documented, executable on just about any computing platform, open source, focused, functional, and at the same time does not try to be all things to all people. Using D2RQ it is more than possible to quickly and easily publish a well-designed relational database as RDF. The process is relatively simple:

* download the software
* use a command-line utility to map the database structure to a configuration file

  * season the configuration file to taste
  * run the D2RQ server using the configuration file
    as input thus allowing people or RDF user-agents
    to search and browse the database using linked
    data principles
  * alternatively, dump the contents of the database
    to an RDF serialization and upload the result
    into your favorite RDF triple store

The downside of D2RQ is its generic nature. It will create an RDF ontology whose terms correspond to the names of database fields. These field names do not map to widely accepted ontologies and therefore will not interact well with communities outside the ones using a specific database structure. Still, the use of D2RQ is quick, easy, and accurate.

The second alternative requires community effort and coordination. The databases of Archivist's Toolkit, Archon, ArchivesSpace, or Past Perfect could be assumed. The community could then get together and decide on an RDF ontology to use for archival descriptions. The database structure(s) could then be mapped to this ontology. Next, programs could be written against the database(s) to create serialized RDF thus beginning the process of publishing linked data. Once that was complete, the archival community would need to come together again to ensure it uses as many shared URIs as possible thus creating the most functional sets of linked data. This second alternative requires a significant amount of community involvement and wide-spread education. It represents a never-ending process.

[1] D2RQ - http://d2rq.org

6. On Your Way: Next Steps

6.a. Integration into daily practice

6.b. Three Cs: Cleanup, Conversion, Consistency

The article entitled Recipes for Enhancing Digital Collections with Linked Data
by Thomas Johnson and Karen Estlund (http://journal.code4lib.org/articles/9214) outlines a number of ways of cleaning up data in content management systems by way of RDF statements.

clean up steps include:

1. Remove noise
2. Normalize presentation
3. Assign URIs for curation objects
4. Map legacy elements to Linked Data vocabularies

As stated by Hillman, the process of moving to linked data is The key to this aug- mentation process involves changing the basic metadata unit from “record” to “statement.” –
http://dcpapers.dublincore.org/pubs/article/view/770/766

Problems with data, again from hillman an:

1. missing data – metadata elements not present in supplied metadata
2. incorrect data – metadata values not con- forming to standard element use
3. confusing data – multiple values crammed into a single metadata element, embedded html tags, etc.
4. insufficient data – e.g., no indication of controlled vocabularies used

Safe transformations include:

1. remove “noise” – a partial solution to the “incorrect data” problem. For example, we remove metadata with no information value, such as empty metadata elements, metadata elements with values such as “unknown” or “n/a” or consisting entirely of dashes or other punctuation.

2. detect and identify controlled vocabular- ies in use whenever possible – a partial solution to the “insufficient data” prob- lem. For example, the DCMIType encod- ing scheme is applied to DC “Type” elements when their value is one of the allowed DCMITypes [10]. This works well for small controlled vocabularies; however, it does not scale well to large vocabularies such as LCSH.

3. normalize metadata presentation – clean up the values: remove double XML en- codings (“<” becomes “<”), extra whitespace (a tab followed by five spaces becomes a single space), etc.

Creating and maintaining metadata is a never-ending process. The items being described can always use elaboration. Collections may increase is size. Rights applied against content may change. Things become digitized, or digitized things are migrated from one format to another. Because of these sorts of things and many others, cleanup, conversion, and consistency are something every metadata specialist needs to keep in mind.

Cleanup, conversion, and consistency means many things. Does all of your metadata use the same set of one or more vocabularies? Are things spelled correctly? Maybe you used abbreviations in one document but spelled things out in another? Have you migrated your JPEG images to JPEG2000 or TIFF formats? Maybe the EAD DTD has been updated, and you want (need) to migrate your finding aids from one XML format to another? Do all of your finding aids exhibit the same level of detail; are some “thinner” than others? Have you used one form of a person’s name in one document but used another form in a different document? The answers to these sorts of questions point to the need for cleanup, conversion, and consistency.

7.a.ii. Gaps: What is needed

There needs to be easy to use tools to find URIs and insert them in to archival descriptions. One such tool is called lobid:

> In “From strings to things: A linked data API for library hackers and Web developers” Fabian Steeg and Pascal Christoph (HBZ) described an interface allowing librarians to determine the URIs of people, places, and things for library catalog records. “How can we benefit from linked data without being linked data experts? We want to pub Web developers into focus using JSON for HTTP.” There are few hacks illustrating some of their work on Github in the lobid repository. --https://github.com/lobid

Another example would be an interface to the varius linked data sets available from the Library of Congress. --http://id.loc.gov

Listed in no priority order, some of the things needed, include:

* hands-on training
* desktop tools enabling people or machines to associate strings with URIs
* a simple RDF statement editor
* the killer app / additional demonstration applications
* a conceptional shift from document to statement

A "real" RDF editing tool is a gap.

A [insert your favorite tool here, such as Archivist's Toolkit, Archon, ArchiveSpace, etc.] to RDF publishing system tool to a gap.

Write "add-ons" to existing systems that output to CIDOC Conceptual Reference Model (CRM)

4.a. Projects: Brief descriptions with an emphasis on tangible benefits and outcomes of each (done)

While the number of linked data websites is less than the worldwide total number, it is really not possible to list every linked data project but only things that will presently useful to the archivist and computer technologist working in cultural heritage institutions. And even then the list of sites will not be complete. Instead, listed below are a number of websites of interest today.

The list is divided into two parts: introductions, data sets and projects. The introductions are akin to directories or initial guilds filled with pointers to information about RDF especially meaningful to archivists (and other cultural heritage workers). The data sets and projects range from simple RDF dumps to full-blown discovery systems. In between some simple browsable lists and raw SPARQL endpoints.

Introductions

* Datahub (http://datahub.io/) - This is a directory of data sets. It includes descriptions of hundreds of data collections. Some of them are linked data sets. Some of them are not.

* LODLAM (http://lodlam.net/) - LODLAM is an acronym for Linked Open Data in Libraries Archives and Museums. LODLAM.net is a community, both virtual and real, of linked data aficionados in cultural heritage institutions. It, like OpenGLAM, is a good place to discuss linked data in general.

* OpenGLAM http://openglam.org) - GLAM is an acronym for Galleries, Libraries, Archives, and Museums. OpenGLAM is a community fostered by the Open Knowledge Foundation and a place to to discuss linked data that is "free". for It, like LODLAM, is a good place to discuss linked data in general.

* semanticweb.org (http://semanticweb.org) - semanticweb.org is a portal for publishing information on research and development related to the topics Semantic Web and Wikis. Includes data.semanticweb.org and data.semanticweb.org/snorql.

Data sets and projects

* 20th Century Press Archives (http://zbw.eu/beta/p20) - This is an archive of digitized newspaper articles which is made accessible not only as HTML but a number of other metadata formats such as RDFa, METS/MODS and OAI-ORE. It is a good example of how metadata publishing can be mixed and matched in a single publishing system.

* AGRIS (http://agris.fao.org/openagris/) - Here you will find a very large collection of bibliographic information from the field of agriculture. It is accessible via quite a number of methods including linked data.

* D2R Server for the CIA Factbook (http://wifo5-03.informatik.uni-mannheim.de/factbook/) - The content of the World Fact Book distributed as linked data.

* D2R Server for the Gutenberg Project (http://wifo5-03.informatik.uni-mannheim.de/gutendata/) - This is a data set of Project Gutenburgh content -- a list of digitized public

domain works, mostly books.

* Dbpedia (http://dbpedia.org/About) - In the simplest terms, this is the content of Wikipedia made accessible as RDF.

* LIBRIS (http://libris.kb.se) - This is the joint catalog of the Swedish academic and research libraries. Search results are presented in HTML, but the URLs pointing to individual items are really actionable URIs resolvable via content negotiation, thus support distribution of bibliographic information as RDF. This initiative is very similar to OpenCat.

* Linked Archives Hub Test Dataset (http://data.archiveshub.ac.uk) - This data set is RDF generated from a selection of archival finding aids harvested by the Archives Hub in the United Kingdom.

* Linked Movie Data Base (http://linkedmdb.org/) - A data set of movie information.

* Linked Open Data at Europeana (http://pro.europeana.eu/datasets) - A growing set of RDF generated from the descriptions of content in Europeana.

* Linking Lives (http://archiveshub.ac.uk/linkinglives/) - While this project has had no working interface, it is a good read on the challenges of presenting link data people (as opposed to computers). Its blog site enumerates and discusses issues from provenance to unique identifiers, from data clean up to interface design.

* LOCAH Project (http://archiveshub.ac.uk/locah/) - This is/was a joint project between Mimas and UKOLN to make Archives Hub data available as structured Linked Data. (All three organizations are located in the United Kingdom.). EAD files were aggregated. Using XSLT, they were transformed into RDF/XML, and the RDF/XML was saved in a triple store. The triple store was then dumped as a file as well as made searchable via a SPARQL endpoint.

* OCLC Data Sets & Services (http://www.oclc.org/data/) - Here you will find a number of freely available bibliographic data sets and services. Some are available as RDF and linked data. Others are Web services.

* OpenCat (http://demo.cubicweb.org/opencatfresnes/) - This is a library catalog combining the authority data (available as RDF) provided by the National Library of France with works of a second library (Fresnes Public Library). Item level search results have URIs whose RDF is available via content negotiation. This project is similar to LIBRIS.

* PELAGIOS (http://pelagios-project.blogspot.com/p/about-pelagios.html) - A data set of ancient places.

* ReLoad (http://labs.regesta.com/progettoReload/en) - This is a collaboration between the Central State Archive of Italy, the Cultural Heritage Institute of Emilia Romagna Region, and Regesta.exe. It is the aggregation of EAD files from a number of archives which have been

transformed into RDF and made available as linked data. Its purpose and intent are very similar to the the purpose and intent of the combined LOCAH Project and Linking Lives.

* World Bank Linked Data (http://worldbank.270a.info/.html) - A data set of World Bank indicators, climate change information, finances, etc.

7. Tools and Visualizations

This section lists various tools for archivists and computer technologists wanting to participate in various aspects of linked data. Here you will find pointers to creating, editing, storing, publishing, and searching linked data.

Directories

The sites listed in this section enumerate linked data and RDF tools. They are jumping off places to other sites:

* ConverterToRdf (http://www.w3.org/wiki/ConverterToRdf) - Hosted by the W3C this is a long list links pointing to applications that will convert varius forms of data into RDF. The list is long, and defunct tools may not be updated as such.

* Linked Data Tools (http://linkeddata.org/tools) - Linkeddata.org is the home of the famous linked data "cloud" image, and LinkedData.org also includes a number of directory-like pages on linked data in general. One of those pages is a list of linked data tools.

* SemWebClients (http://www.w3.org/wiki/TaskForces/CommunityProjects/LinkingOpenData/SemWebClients) - Hosted by the W3C, this is a list of Semantic Web clients and applications.

* RDFImportersAndAdapters (http://www.w3.org/wiki/RDFImportersAndAdapters) - A list of tools and applications for converting various data and file formats into serialized RDF.

* SIMILE RDFizers (http://simile.mit.edu/wiki/RDFizers) - Another list of tools for converting data and files into RDF.

* SparqlImplementations (http://www.w3.org/wiki/SparqlImplementations) - Another page hosted by the W3C, and listing SPARQL clients and tools.

* Tools (http://www.w3.org/2001/sw/wiki/Tools) - Hosted by the W3C, this page is a sort of directory of directories for all sort of linked data and RDF tools: editors, converters, SPARQL clients, triple stores, etc.

RDF converters, validators, etc.

* ead2rdf (http://data.archiveshub.ac.uk/xslt/ead2rdf.xsl) - This is the XST stylesheet previously used by the Archives Hub in their LOCAH Linked Archives Hub project. It transforms EAD files into RDF/XML. A slightly modified version of this stylesheet was used to create the LiAM "sandbox".

* Protégé (http://protege.stanford.edu) - Install this well-respected tool locally or use it as a hosted Web application to create OWL ontologies.

* RDF2RDF (http://www.l3s.de/~minack/rdf2rdf/) - A handy Java jar file enabling you to convert various versions of serialized RDF into other

versions of serialized RDF.

* Vapour, a Linked Data Validator (http://validator.linkeddata.org/vapour) - Much like the W3C validator, this online tool will validate the RDF at the other end of a URI. Unlike the W3C validator, it echoes back and forth the results of the content negotiation process.

* W3C RDF Validation Service (http://www.w3.org/RDF/Validator/) - Enter a URI or paste an RDF/XML document into the text field, and a triple representation of the corresponding data model as well as an optional graphical visualization of the data model will be displayed.

Linked data frameworks and publishing systems

* 4store (http://4store.org/) - A linked data publishing framework for managing triple stores, querying them locally, querying them via SPARQL, dumping their contents to files, as well as providing support via a number of scripting languages (PHP, Ruby, Python, Java, etc.).

* Apache Jena (http://jena.apache.org/) - This is a set of tools for creating, maintaining, and publishing linked data complete a SPARQL engine, a flexible triple store application, and inference engine.

* D2RQ (http://d2rq.org/) - Use this application to provide a linked data front-end to any (well-designed) relational database. It supports SPARQL, content negotiation, and RDF dumps for direct HTTP access or uploading into triple store.

* oai2lod (https://github.com/behas/oai2lod) - This is a particular implementation D2RQ Server. More specifically, this tool is an intermediary between a OAI-PMH data providers and a linked data publishing system. Configure oai2lod to point to your OAI-PMH server and it will publish the server's metadata as linked data.

* OpenLink Virtuoso Open-Source Edition (https://github.com/openlink/virtuoso-opensource/) - An open source version of OpenLink Virtuoso. Feature-rich and well-documented.

* OpenLink Virtuoso Universal Server (http://virtuoso.openlinksw.com) - This is a commercial version of OpenLink Virtuoso Open-Source Edition. It seems to be a platform for modeling and accessing data in a wide variety of forms: relational databases, RDF triples stores, etc. Again, feature-rich and well-documented.

* openRDF (http://www.openrdf.org/) - This is a Java-based framework for implementing linked data publishing including the establishment of a triple store and a SPARQL endpoint.

* openRDF (http://www.openrdf.org/) - This is a Java-based framework for implementing linked data publishing including the establishment of a triple store and a SPARQL endpoint.

* R2RML (http://www.w3.org/TR/r2rml/) -

Semantic Web browsers

This is a small set of Semantic Web browsers. Give them URIs and they allow you to follow and describe the links they include.

* Inspector (http://inspector.sindice.com) - Creates graphs, lists ontologies, generates a full-text report from literals, and lists triples found at the other end of URIs.

* LinkSailor (http://linksailor.com) - Feed this interface a URI that points to many other actionable URIs, and it will return an interesting mash-up including images, paragraphs of texts, and hyperlinks to more content.

* LOD Browser Switch (http://browse.semanticweb.org) - This is really a gateway to other Semantic Web browsers. Feed it a URI and it will create lists of URLs pointing to Semantic Web interfaces, but many of the URLs (Semantic Web interfaces) do not seem to work. Some of the resulting URLs point to RDF\ serialization converters

* LodLive (http://en.lodlive.it) - This Semantic Web browser allows you to feed it a URI and interactively follow the links associated with it. URIs can come from DBedia, Freebase, or one of your own.

* Open Link Data Explorer (http://demo.openlinksw.com/rdfbrowser2/) - The most sophisticated Semantic Web browser in this set. Given a URI it creates various views of the resulting triples associated with including lists of all its properties and objects, networks graphs, tabular views, and maps (if the data includes geographic points).

* Quick and Dirty RDF browser (http://graphite.ecs.soton.ac.uk/browser/) - Given the URL pointing to a file of RDF statements, this tool returns all the triples in the file and verbosely lists each of their predicate and object values. Quick and easy. This is a good for reading everything about a particular resource. The tool does not seem to support content negotiation.

If you need some URIs to begin with, then try some of these:

* Ray Family Papers - http://infomotions.com/sandbox/liam/data/mum432.rdf
* Catholics and Jews - http://infomotions.com/sandbox/liam/data/shumarc681792.rdf
* Walt Disney via VIAF - http://viaf.org/viaf/36927108/
* origami via the Library of Congress - http://id.loc.gov/authorities/subjects/sh85095643
* Paris from DBpedia - http://dbpedia.org/resource/Paris

Directories of ontologies

* Linked Open Vocabularies (http://lov.okfn.org/dataset/lov/) - This is a list of more than 400 RDF ontologies with both a searchable/browsable interface. It is als accessible via linked data publishing principles.

* Open Metadata Registry (http://metadataregistry.net) - This is a list of more than 300 RDF ontologies. The Registry was originally conceived of as a National Science Foundation project called the National Science Foundation Digital Library (NSDL) Registry. The Registry seems heavy on sets of controlled vocabularies of a bibliographic nature

Vocabularies

* Getty Vocabularies (http://vocab.getty.edu) - A set of thesari used to "categorize, describe, and index cultural heritage objects and information".

* Library of Congress Linked Data Service (http://id.loc.gov/) - A set of data sets used for bibliographic classification: subjects, names, genres, formats, etc.

* New York Times (http://data.nytimes.com/) - A list of New York Times subject headings.

* VIAF (http://viaf.org/) - This data set functions as a name authority file. The human-readable interface is quite nice.

Ontologies

* Archival Collections Ontology (https://github.com/rubinsztajn/archival) - Created by Aaron Rubinstein (University of Massachusetts) is small ontology specifically designed for archival collections.

* Bibo (http://purl.org/ontology/bibo/) - Just as the name infers, this ontology is intended to model bibliographic citations.

* Bibframe (http://bibframe.org) - Another ontology which is bibliographic in nature and includes an abundance of properties for various identifiers.

* CIDOC Conceptual Reference Model (http://www.cidoc-crm.org) - Intended for cultural heritage institutions, this model includes quite a number of subclass items surrounding two primary concepts: things and time.

* Dublin Core Terms (http://dublincore.org/documents/dcmi-terms/) - This ontology is rather bibliographic in nature, and provides a framework for describing many of ht of archival descriptions.

* Europeneana (http://ontogenealogy.com/europeana-data-model-edm/) - Another approach is to the data model from other oganizations. Since Europeneana's data is intended to be available as linked data, then it might be a good model to explore – http://pro.europeana.eu/edm-documentation Specifically: For the archival community, collection level descriptions such as EAD play a major role.They fit neatly under the EDM,in particular the notion of ore:aggregration allows for describing archival "fonds". The International Council of Archives just started the discussion about a common conceptual model similar to FRBR or the CRM. In the meanwhile, with the CRM historical facts associated with archival contents can be described in more detail than just on the EDM level (Stasinopoulou et. al. 2007). Further, collection-level

descriptions in Dublin Core are quite convenient and becoming popular for archival descriptions. –

  * FOAF (http://www.foaf-project.org/) - Friend Of A Friend (FOAF) is a popular and often-quoted ontology for modeling people and the social networks they create: names, email address, social network account identifiers, etc.

  * LOCAH RDF Vocabulary (http://data.archiveshub.ac.uk/def/) - This is the model designed for the LOCAH Project from the Archives Hub in the United Kingdom. It is the model underpinning the transformed EAD files of the LiAM "sandbox".

  * MARC21 Vocabularies from Metadata Management Associates (http://marc21rdf.info) -

  * OAD Vocabulary - (http://labs.regesta.com/progettoReload/wp-content/uploads/2013/08/oadNew.html) - "[Google translation] The definition of an ontology of archival description is needed in order to test the potential of the web of data to archival descriptions. The archival description aims at the representation of a unit of description by collecting, analyzing, organizing and recording the information needed to identify, manage, locate and explain the context and documentary material and the storage systems that produced (ISAD (G)). The ontology of archival description (SRO) has as its goal the formal representation of the descriptions of the individual units of description - understood as objects of their archival descriptions. In particular, the SRO waiver to take into account all the individual elements of these descriptions, in endless variations they present as part of the archival systems in which they are hinged, but instead seeks only to explain the elements of information deemed necessary for the exposure the web of data units of archival description to ensure integration with other datasets published also in format Linked Open Data. SROs, in beta, is expressed in OWL (Ontology Web Language): it takes into account all the elements of archival description in the standard ISAD (G): general international standard archival description, adopted by the ICA (International Council on Archives) integrating them with other information elements not covered by the standard mentioned - as the index entries - and with links to creators and conservative. The formal mechanisms provided by the standard RDF and OWL have made it possible to bring the information elements of archival descriptions expressed in an ontology SROs to "external" concepts representative of traditional archival descriptive and based on the international standard ISAD (G). Since the experiments conducted by the partners of the project ReLoad had as object data encoded on the basis of the framework EAD (Encoded Archival Description), the ontology provides for a specific class OAD "eadElement" designed to encode the information on the element or attribute of the EAD scheme used by the organization that provides data archival description, which is also attributed to the ISAD (G) on the basis of official mapping ISAD (G) - EAD

  * OWL (http://www.w3.org/2001/sw/wiki/OWL) -

  * PROV (http://www.w3.org/TR/prov-overview/) - for provenance information.

  * RDF - This ontology is necessary because linked data is manifested as... RDF

  * RDFS (http://www.w3.org/TR/rdf-schema/) - This ontology may be necessary because the archival community may be creating some of its own ontologies.

  * Schema.org (http://schema.org) - Originally sponsored by Google, Microsoft, Yahoo, etc. this ontology is advocated by the major Internet search engines. It is a growing, well-supported ontology, and it is seemingly designed for just about anything under the sun.

  * SKOS (http://www.w3.org/2004/02/skos/) - This ontology is used to create thesauri.

  * VoID (http://semanticweb.org/wiki/VoID) - This relatively simple data model is used to describe RDF data sets with things like their location, size, licensing terms, frequency of updates, etc.

Appendix A: content-negotiation and cURL

This is the tiniest of introductions to content negotiation and cURL.

The computer technology behind linked data is about two things: 1) serializing RDF, and 2) making it available on the Web. Various RDF serializations are described in another section of the Guidebook. The second thing, making RDF available on the Web, can be accomplished in any number of ways, including but not necessarily limited to: 1) embedded in HTML as RDFa, 2) "dumps" of RDF, 3) SPARQL interfaces, and 3) content negotiation. Exploiting RDFa was discussed in a previous section. A number of the data sets in Projects section make their RDF available as "dumps". The next section of the Guidebook is a tutorial on SPARQL. This section describes content negotiation and a command line tool called cURL, which is very helpful for understanding content negotiation.

Content negotiation is an HTTP-pure technique for exchanging data on the Web. In the briefest of descriptions, content negotiation is a client-server technique where the client application first requests some data via a URI in a specific format (plain text, HTML, PDF, RDF/XML, etc.). The server responses with either a "file not found" error, or a URL where the request can be satisfied. It is then up to the client to make a second request with the given URL to obtain the desired data. Content negotiation and the complementary "REST-ful" computing are the primary means of Web-based data exchange, and it is interesting to note the differences between the two. Content negotiation only requires an (in-depth) knowledge of HTTP to implement. Given a URI and a thorough knowledge of HTTP, a programmer can effectively harvest linked data. On the other hand, REST-ful interfaces, while requiring less knowledge of HTTP, are often specific to individual websites. They also often require API "keys" as well as the use of very long URLs complete with domain-specific name/value pairs. Content negotiation is more standards-based when compared to REST-ful computing, but REST-ful computing is easier to initially grasp. Both have their advantages and disadvantages, but content negotiation is the way of linked data.

CURL is a command-line tool making it easier for you to see the Web as data and not presentation. Consequently it is a ver good tool for learning about content negotiation. Please don't be afraid of cURL because it is a command-line utility. Understanding how to use cURL and to do content-negotiation by hand will take you a long way in understanding linked data.

The first step is to download and install cURL. If you have a Macintosh or a Linux computer, then it is probably already installed. If not, then give the cURL download wizard a whirl. [1] We'll wait.

Next, you need to open a terminal. On Macintosh computers a terminal application is located in the Utilities folder of your Applications folder. It is called "Terminal". People using Windows-based computers can find the "Command" application by searching for it in the Start Menu. Once cURL has been installed and a terminal has been opened, then you can type the following command at the prompt to display a help text:

```
curl --help
```

There are many options there, almost too many. It is often useful to view only one page of text at a time, and you can "pipe" the output through to a program called "more" to do this:

```
curl --help | more
```

By pressing the space bar, you can go forward in the display. By pressing "b" you can go backwards, and by pressing "q" you can quit.

Feed cURL the complete URL of Google's home page to see how much content actually goes into their "simple" presentation:

```
curl http://www.google.com/ | more
```

The communication of the World Wide Web (the hypertext transfer protocol or HTTP) is divided into two

parts: 1) a header, and 2) a body. By default cURL displays the body content. To see the header, add the -I (for a mnemonic, think "information") to the command:

```
curl -I http://www.google.com/
```

The result will be a list of characteristics the remote Web server is using to describe this particular interaction between itself and cURL. The most important things to note are: 1) the status line and 2) the content type. The status line will be the first line in the result, and it will say something like "HTTP/1.1 200 OK", meaning there were no errors. Another line will begin with "Content-Type:" and denotes the format of the data being transferred. In most cases the content type line will include something like "text/html" meaning the content being sent is in the form of an HTML document.

Now feed cURL a URI for Walt Disney, such as one from DBpedia:

```
curl http://dbpedia.org/resource/Walt_Disney
```

The result will be empty, but upon the use of the -I switch you can see how the status line changed to "HTTP/1.1 303 See Other". This means there is no content at the given URI, and the line starting with "Location:" is a pointer – an instruction – to go to a different document. In the parlance of HTTP this is called redirection. Using cURL going to the recommended location results in a stream of HTML:

```
curl http://dbpedia.org/page/Walt_Disney | more
```

Most Web browsers automatically follow HTTP redirection commands, but cURL needs to be told this explicitly through the use of the -L switch. (Think "location".) Consequently, given the original URI, the following command will display HTML even though the URI has no content:

```
curl -L http://dbpedia.org/resource/Walt_Disney | more
```

Now remember, the Semantic Web and linked data depend on the exchange of RDF, and upon closer examination you can see there are "link" elements in the resulting HTML, and these elements point to URLs with the .rdf extension. Feed these URLs to cURL to see an RDF representation of the Walt Disney data:

```
curl http://dbpedia.org/data/Walt_Disney.rdf | more
```

Downloading entire HTML streams, parsing them for link elements containing URLs of RDF, and then requesting the RDF is not nearly as efficient as requesting RDF from the remote server in the first place. This can be done by telling the remote server you accept RDF as a format type. This is accomplished through the use of the -H switch. (Think "header".) For example, feed cURL the URI for Walt Disney and specify your desire for RDF/XML:

```
curl -H "Accept: application/rdf+xml" http://dbpedia.org/resource/Walt_Disney
```

Ironically, the result will be empty, and upon examination of the HTTP headers (remember the -I switch) you can see that the RDF is located at a different URL, namely, http://dbpedia.org/data/Walt_Disney.xml:

```
curl -I -H "Accept: application/rdf+xml" http://dbpedia.org/resource/Walt_Disney
```

Finally, using the -L switch, you can use the URI for Walt Disney to request the RDF directly:

```
curl -L -H "Accept: application/rdf+xml" http://dbpedia.org/resource/Walt_Disney
```

That is cURL and content-negotiation in a nutshell. A user-agent submits a URI to a remote HTTP server and specifies the type of content it desires. The HTTP server responds with URLs denoting the location of the desired content. The user-agent then makes a more specific request. It is sort of like the movie. "One URI to rule them all." In summary, remember:

  * cURL is a command-line user-agent
  * given a URL, cURL returns, by default, the body of an HTTP transaction
  * the -I switch allows you to see the HTTP header
  * the -L switch makes cURL automatically follow HTTP redirection requests
  * the -H switch allows you to specify the type of content you wish to accept
  * given a URI and the use of the -L and -H switches you are able to retrieve either HTML or RDF

Use cURL to actually see linked data in action, and here are a few more URIs to explore:

  * Walt Disney via VIAF - http://viaf.org/viaf/36927108/
  * origami via the Library of Congress - http://id.loc.gov/authorities/subjects/sh85095643
  * Paris from DBpedia - http://dbpedia.org/resource/Paris

[1] cURL download wizard - http://curl.haxx.se/dlwiz/

10 - SPARQL tutorial (done)

This is the simplest of SPARQL tutorials. The tutorial's purpose is two-fold: 1) through a set of examples, introduce the reader to the syntax of SPARQL queries, and 2) to enable the reader to initially explore any RDF triple store which is exposed as a SPARQL endpoint.

SPARQL (SPARQL protocol and RDF query language) is a set of commands used to search RDF triple stores. It is modeled after SQL (structured query language), the set of commands used to search relational databases. If you are familiar with SQL, then SPARQL will be familiar. If not, then think of SPARQL queries as formalized sentences used to ask a question and get back a list of answers.

Also, remember, RDF is a data structure of triples: 1) subjects, 2) predicates, and 3) objects. The subjects of the triples are always URIs -- identifiers of "things". Predicates are also URIs, but these URIs are intended to denote relationships between subjects and objects. Objects are preferably URIs but they can also be literals (words or numbers). Finally, RDF objects and predicates are defined in human-created ontologies as a set of classes and properties where classes are abstract concepts and properties are characteristics of the concepts.

Try the following steps with just about any SPARQL endpoint:

1. Get an overview - A good way to begin is to get a list of all the ontological classes in the triple store. In essence, the query below says, "Find all the unique objects (classes) in the triple store where any subject is a type of object, and sort the result by object."

```
SELECT DISTINCT ?o WHERE { ?s a ?o } ORDER BY ?o
```

2. Learn about the employed ontologies - Ideally, each of the items in the result will be an actionable URI in the form of a "cool URL". Using your Web browser, you ought to be able to go to the URL and read a thorough description of the given class, but the URLs are not always actionable.

3. Learn more about the employed ontologies - Using the following query you can create a list of all the properties in the triple store as well as infer some of the characteristics of each class. Unfortunately, this particular query is intense. It may require a long time to process or may not return at all. In English, the query says, "Find all the unique predicates where the RDF triple has any subject, any predicate, or any object, and sort the result by predicate."

```
SELECT DISTINCT ?p WHERE { ?s ?p ?o } ORDER BY ?p
```

4. Guess - Steps #2 and Step #3 are time intensive, and consequently it is sometimes easier just browse the triple store by selecting one of the "cool URLs" returned in Step #1. Submit a modified version of Step #1's query. It says, "Find all the subjects where any RDF subject (URI) is a type of object (class)". Using the LiAM triple store, the following query tries to find all the things that are EAD finding aids.

```
SELECT ?s WHERE { ?s a <http://data.archiveshub.ac.uk/def/FindingAid> }
```

5. Learn about a specific thing - The result of Step #4 ought to be a list of (hopefully actionable) URIs. You can learn everything about that URI with the following query. It says, "Find all the predicates and objects in the triple store where the RDF triple's subject is a given value and the predicate and object are of any value, and sort the result by predicate". In this case, the given value is one of the items returned from Step #4.

```
SELECT ?p ?o WHERE { <http://infomotions.com/sandbox/liam/id/mum432> ?p ?o } ORDER BY ?p
```

6. Repeat a few times - If the results from Step #5 returned seemingly meaningful and complete information about your selected URI, then repeat Step #5 a few times to get a better feel for some of the "things" in the triple store. If the results were not meaningful, then got to Step #4 to browser another class.

7. Take these hints - The first of these following two queries generates a list of ten URIs pointing to things that came from MARC records. The second query is used to return everything about a specific URI whose data came from a MARC record.

```
SELECT ?s WHERE { ?s a <http://simile.mit.edu/2006/01/ontologies/mods3#Record> } LIMIT 10
SELECT ?p ?o WHERE { <http://infomotions.com/sandbox/liam/id/shumarc681792> ?p ?o } ORDER BY ?p
```

8. Read the manual - At this point, it is a good idea to go back to Step #2 and read the more formal descriptions of the underlying ontologies.

9. Browse some more - If the results of Step #3 returned successfully, then browse the objects in the triple store by selecting a predicate of interest. The following queries demonstrate how to list things like titles, creators, names, and notes.

```
SELECT ?s ?o WHERE { ?s <http://purl.org/dc/terms/title> ?o } ORDER BY ?o LIMIT 100
SELECT ?s ?o WHERE { ?s <http://simile.mit.edu/2006/01/roles#creator> ?o } ORDER BY ?o LIMIT 100
SELECT ?s ?o WHERE { ?s <http://xmlns.com/foaf/0.1/name> ?o } ORDER BY ?o LIMIT 100
SELECT ?s ?o WHERE { ?s <http://data.archiveshub.ac.uk/def/note> ?o } ORDER BY ?o LIMIT 100
```

10. Read about SPARQL - This was the tiniest of SPARQL tutorials. Using the LiAM data set as an example, it demonstrated how to do the all but simplest queries against an RDF triple store. There is a whole lot more to SPARQL than SELECT, DISTINCT, WHERE, ORDER BY, and LIMIT commands. SPARQL supports a short-hand way of denoting classes and properties called PREFIX. It supports Boolean operations, limiting results based on "regular expressions", and a few mathematical functions. SPARQL can also be used to do inserts and deletes against triple stores. The next step is to read more about SPARQL. Consider reading the canonical documentation from the W3C, "SPARQL by example", and the Jena project's "SPARQL Tutorial". [1, 2, 3]

Finally, don't be too intimidated about SPARQL. Yes, it is possible to submit SPARQL queries by hand, but in reality, person-friendly front-ends are expected to be created making search much easier.

[1] canonical documentation - http://www.w3.org/TR/rdf-sparql-query/
[2] SPARQL By Example - https://www.cambridgesemantics.com/semantic-university/sparql-by-example
[3] SPARQL Tutorial - http://jena.apache.org/tutorials/sparql.html

03.g Glossary - This is a beginner's glossary to linked data. (done)

  * API - (see application programmer interface)

  * application programmer interface (API) - an abstracted set of functions and commands used to get output from remote computer applications. These functions and commands are not necessarily tied to any specific programming language and therefore allow programmers to use a programming language of their choice.

  * cool URL - a relatively short, human-readable pointer to Internet accessible content. Cool URLs are expected to be constant, in that they don't change. Additionally and in general, cool URLs do not include question marks (?) nor name/value pairs denoting queries.

  * content negotiation - a process whereby a user-agent and HTTP server mutually decide what data format will be exchanged during an HTTP request. In the world of linked data, content negotiation is very important when URIs are requested because content negotiation helps determine whether or not HTML or serialized RDF will be returned.

  * extensible markup language (XML) - a standardized data structure made up of a minimum of rules and can be easily used to represent everything from tiny bits of data to long narrative texts. XML is designed to be read my people as well as computers, but because of this it is often considered verbose, and ironically, difficult to read.

  * file transfer protocol (FTP) - A Internet standard for copying files from one Internet host to another.

  * FTP - (see file transfer protocol)

  * HTML - (see hypertext markup language)

  * HTTP - (see hypertext transfer protocol)

  * hypertext markup language (HTML) - an XML-like data structure intended to be rendered by user-agents whose output is for people to read. For the most part, HTML is used to markup text and denote a text's stylistic characteristics such as headers, paragraphs, and list items. It is also used do markup the hypertext links (URLs) between documents.

  * hypertext transfer protocol (HTTP) - the formal name for the way the World Wide Web operates. It begins with one computer program (a user-agent) requesting content from another computer program (a server) and getting back a response. Once received, the response is formatted for reading or for processing by a computer program. The shape and content of both the request and the response are what make-up the protocol.

  * Javascript object notation (JSON) - like XML, a data structure allowing arbitrarily large sets of values to be associated with an arbitrarily large set of names (variables). JSON was first natively implemented as a part of the Javascript computer language, but has since become popular in other computer languages as well.

  * JSON - (see Javascript object notation)

  * linked data - the content and technical process for making real the ideas behind the Semantic Web. It begins with the creation of serialized RDF and making the serialization available via HTTP. User agents are then expected to harvest the RDF, combine it with other harvested RDF, and ideally use it to bring to light new or existing relationships between real world objects -- people, places, and things -- thus creating and enhancing human knowledge.

  * linked open data - a qualification of linked data whereby the information being exchanged is expected

to be "free" as in gratis.

* OAI - (see Open Archives Initiative-Protocol for Metadata Harvesting)

* ontology - a highly structured vocabulary, and in the parlance of linked data, used to denote, describe, relate, and qualify the predicates of RDF triples. Ontologies have been defined for a very wide range of human domains, everything from bibliography (Dublin Core or MODS), to people (FOAF), to sounds (Audio Features).

* Open Archives Initiative-Protocol for Metadata Harvesting (OAI-PMH) - a metadata publishing standard consisting of a set commands whereby information is listed, requested, and exchanged between two computers in the Internet. OAI-PMH is complementary to the principles and practices of linked data.

* RDF - (see resource description framework)

* representational state transfer (REST) - a process for querying remote HTTP servers and getting back computer-readable results. The process usually employs denoting name-value pairs in a URL and getting back something like XML or JSON.

* resource description framework - the conceptual model for describing the knowledge of the Semantic Web. It is rooted in the notion of triples whose subjects and objects are literally linked with other triples through the use of URIs.

* REST - (see representational state transfer)

* Semantic Web - an idea articulated by Tim Berners Lee whereby human knowledge is expressed in a computer-readable fashion and made available via HTTP so computers can harvest it and bring to light new information or knowledge.

* serialization - a manifestation of RDF; one of any number of textual expressions of RDF triples. Examples include but are not limited to RDF/XML, RDFa, N3, and JSON-LD.

* SPARQL - (see SPARQL protocol and RDF query language)

* SPARQL protocol and RDF query language (SPARQL) - a formal specification for querying and returning results from RDF triple stores. It looks and operates very much like the structured query language (SQL) of relational databases complete with its SELECT, WHERE, and ORDER BY clauses.

* triple - the atomistic facts making up RDF. Each fact is akin to a rudimentary sentence with three parts: 1) subject, 2) predicate, and 3) object. Subjects are expected to be URIs. Ideally, objects are URIs as well, but can also be literals (words, phrases, or numbers). Predicates are akin to the verbs in a sentence and they denote a relationship between the subject and object. Predicates are expected to be a member of a formalized ontology.

* triple store - a database of RDF triples usually accessible via SPARQL

* universal resource identifier (URI) - a unique pointer to a real-world object or a description of an object. In the parlance of linked data, URIs are expected to have the same shape and function as URLs, and if they do, then the URIs are often described as "actionable".

* universal resource locator (URL) - an address denoting the location of something on the Internet. These addresses usually specify a protocol (like http), a host (or computer) where the protocol is implemented, and a path (directory and file) specifying where on the computer the item of interest resides.

* URI - (see universal resource identifier)

  * URL - (see universal resource locator)

  * user agent - this is the formal name for what is commonly called a "Web browser", but Web browsers usually denote applications where people are viewing the results. User agents are usually "Web browsers" whose readers are computer programs.

  * XML - (see extensible markup language)

For a more complete and exhaustive glossary, see the W3C's Linked Data Glossary. [1]

[1] W3C's Linked Data Glossary - http://www.w3.org/TR/ld-glossary/

8.3 Further reading

This is a list of links and citations to get one started on Linked Open Data

admin. “Barriers to Using EAD,” August 4, 2012. http://oclc.org/research/activities/eadtools.html.

Becker, Christian, and Christian Bizer. “Exploring the Geospatial Semantic Web with DBpedia Mobile.” Web Semantics: Science, Services and Agents on the World Wide Web 7, no. 4 (December 2009): 278–286. doi:10.1016/j.websem.2009.09.004.

Belleau, François, Marc-Alexandre Nolin, Nicole Tourigny, Philippe Rigault, and Jean Morissette. “Bio2RDF: Towards a Mashup to Build Bioinformatics Knowledge Systems.” Journal of Biomedical Informatics 41, no. 5 (October 2008): 706–716. doi:10.1016/j.jbi.2008.03.004.

Berners-Lee, Tim. “Linked Data - Design Issues.” Accessed August 4, 2013. http://www.w3.org/DesignIssues/LinkedData.html.

Berners-Lee, Tim, James Hendler, and Ora Lassila. “The Semantic Web.” Scientific American 284, no. 5 (May 2001): 34–43. doi:10.1038/scientificamerican0501-34.

Bizer, Christian, Tom Heath, and Tim Berners-Lee. “Linked Data - The Story So Far:” International Journal on Semantic Web and Information Systems 5, no. 3 (33 2009): 1–22. doi:10.4018/jswis.2009081901.

Carroll, Jeremy J., Christian Bizer, Pat Hayes, and Patrick Stickler. “Named Graphs.” Web Semantics: Science, Services and Agents on the World Wide Web 3, no. 4 (December 2005): 247–267. doi:10.1016/j.websem.2005.09.001.

“Chem2bio2rdf - How to Publish Data Using D2R?” Accessed January 6, 2014. http://chem2bio2rdf.wikispaces.com/How+to+publish+data+using+D2R%3F.

“Content Negotiation.” Wikipedia, the Free Encyclopedia, July 2, 2013. https://en.wikipedia.org/wiki/Content_negotiation.

“Cool URIs for the Semantic Web.” Accessed November 3, 2013. http://www.w3.org/TR/cooluris/.

Correndo, Gianluca, Manuel Salvadores, Ian Millard, Hugh Glaser, and Nigel Shadbolt. “SPARQL Query Rewriting for Implementing Data Integration over Linked Data.” 1. ACM Press, 2010. doi:10.1145/1754239.1754244.

David Beckett. “Turtle.” Accessed August 6, 2013. http://www.w3.org/TR/2012/WD-turtle-20120710/.

“Debugging Semantic Web Sites with cURL | Cygri’s Notes on Web Data.” Accessed November 3, 2013. http://richard.cyganiak.de/blog/2007/02/debugging-semantic-web-sites-with-curl/.

Dunsire, Gordon, Corey Harper, Diane Hillmann, and Jon Phipps. “Linked Data Vocabulary Management: Infrastructure Support, Data Integration, and Interoperability.” Information Standards Quarterly 24, no. 2/3 (2012): 4. doi:10.3789/isqv24n2-3.2012.02.

Elliott, Thomas, Sebastian Heath, and John Muccigrosso. “Report on the Linked Ancient World Data Institute.” Information Standards Quarterly 24, no. 2/3 (2012): 43. doi:10.3789/isqv24n2-3.2012.08.

Fons, Ted, Jeff Penka, and Richard Wallis. “OCLC’s Linked Data Initiative: Using Schema.org to Make Library Data Relevant on the Web.” Information Standards Quarterly 24, no. 2/3 (2012): 29. doi:10.3789/isqv24n2-3.2012.05.

Hartig, Olaf. “Querying Trust in RDF Data with tSPARQL.” In The Semantic Web: Research and Applications,

edited by Lora Aroyo, Paolo Traverso, Fabio Ciravegna, Philipp Cimiano, Tom Heath, Eero Hyvönen, Riichiro Mizoguchi, Eyal Oren, Marta Sabou, and Elena Simperl, 5554:5–20. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009. http://www.springerlink.com/index/10.1007/978-3-642-02121-3_5.

Hartig, Olaf, Christian Bizer, and Johann-Christoph Freytag. “Executing SPARQL Queries over the Web of Linked Data.” In The Semantic Web - ISWC 2009, edited by Abraham Bernstein, David R. Karger, Tom Heath, Lee Feigenbaum, Diana Maynard, Enrico Motta, and Krishnaprasad Thirunarayan, 5823:293–309. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009. http://www.springerlink.com/index/10.1007/978-3-642-04930-9_19.

Heath, Tom, and Christian Bizer. “Linked Data: Evolving the Web into a Global Data Space.” Synthesis Lectures on the Semantic Web: Theory and Technology 1, no. 1 (February 9, 2011): 1–136. doi:10.2200/S00334ED1V01Y201102WBE001.

Isaac, Antoine, Robina Clayphan, and Bernhard Haslhofer. “Europeana: Moving to Linked Open Data.” Information Standards Quarterly 24, no. 2/3 (2012): 34. doi:10.3789/isqv24n2-3.2012.06.

Kobilarov, Georgi, Tom Scott, Yves Raimond, Silver Oliver, Chris Sizemore, Michael Smethurst, Christian Bizer, and Robert Lee. “Media Meets Semantic Web – How the BBC Uses DBpedia and Linked Data to Make Connections.” In The Semantic Web: Research and Applications, edited by Lora Aroyo, Paolo Traverso, Fabio Ciravegna, Philipp Cimiano, Tom Heath, Eero Hyvönen, Riichiro Mizoguchi, Eyal Oren, Marta Sabou, and Elena Simperl, 5554:723–737. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009. http://www.springerlink.com/index/10.1007/978-3-642-02121-3_53.

LiAM. “LiAM: Linked Archival Metadata.” Accessed July 30, 2013. http://sites.tufts.edu/liam/.

“Linked Data.” Wikipedia, the Free Encyclopedia, July 13, 2013. http://en.wikipedia.org/w/index.php?title=Linked_data&oldid=562554554.

“Linked Data Glossary.” Accessed January 1, 2014. http://www.w3.org/TR/ld-glossary/.

“Linked Open Data.” Europeana. Accessed September 12, 2013. http://pro.europeana.eu/web/guest;jsessionid=09A5D79E7474609AE246DF5C5A18DDD4.

“Linked Open Data in Libraries, Archives, & Museums (Google Group).” Accessed August 6, 2013. https://groups.google.com/forum/#!forum/lod-lam.

“Linking Lives | Using Linked Data to Create Biographical Resources.” Accessed August 16, 2013. http://archiveshub.ac.uk/linkinglives/.

“LOCAH Linked Archives Hub Test Dataset.” Accessed August 6, 2013. http://data.archiveshub.ac.uk/.

“LODLAM - Linked Open Data in Libraries, Archives & Museums.” Accessed August 6, 2013. http://lodlam.net/.

“Notation3.” Wikipedia, the Free Encyclopedia, July 13, 2013. http://en.wikipedia.org/w/index.php?title=Notation3&oldid=541302540.

“OWL 2 Web Ontology Language Primer.” Accessed August 14, 2013. http://www.w3.org/TR/2009/REC-owl2-primer-20091027/.

Quilitz, Bastian, and Ulf Leser. “Querying Distributed RDF Data Sources with SPARQL.” In The Semantic Web: Research and Applications, edited by Sean Bechhofer, Manfred Hauswirth, Jörg Hoffmann, and Manolis Koubarakis, 5021:524–538. Berlin, Heidelberg: Springer Berlin Heidelberg. Accessed September 4, 2013. http://www.springerlink.com/index/10.1007/978-3-540-68234-9_39.

“RDF/XML.” Wikipedia, the Free Encyclopedia, July 13, 2013. http://en.wikipedia.org/wiki/RDF/XML.

“RDFa.” Wikipedia, the Free Encyclopedia, July 22, 2013. http://en.wikipedia.org/wiki/RDFa.

“Semantic Web.” Wikipedia, the Free Encyclopedia, August 2, 2013. http://en.wikipedia.org/w/index.php?title=Semantic_Web&oldid=566813312.

“SPARQL.” Wikipedia, the Free Encyclopedia, August 1, 2013. http://en.wikipedia.org/w/index.php?title=SPARQL&oldid=566718788.

“SPARQL 1.1 Overview.” Accessed August 6, 2013. http://www.w3.org/TR/sparql11-overview/.

“Spring/Summer 2012 (v.24 No.2/3) - National Information Standards Organization.” Accessed August 6, 2013. http://www.niso.org/publications/isq/2012/v24no2-3/.

Summers, Ed, and Dorothea Salo. Linking Things on the Web: A Pragmatic Examination of Linked Data for Libraries, Archives and Museums. ArXiv e-print, February 19, 2013. http://arxiv.org/abs/1302.4591.

“The Linking Open Data Cloud Diagram.” Accessed November 3, 2013. http://lod-cloud.net/.

“The Trouble with Triples | Duke Collaboratory for Classics Computing (DC3).” Accessed November 6, 2013. http://blogs.library.duke.edu/dcthree/2013/07/27/the-trouble-with-triples/.

Tim Berners-Lee, James Hendler, and Ora Lassila. “The Semantic Web.” Accessed September 4, 2013. http://www.scientificamerican.com/article.cfm?id=the-semantic-web.

“Transforming EAD XML into RDF/XML Using XSLT.” Accessed August 16, 2013. http://archiveshub.ac.uk/locah/tag/transform/.

“Triplestore - Wikipedia, the Free Encyclopedia.” Accessed November 11, 2013. http://en.wikipedia.org/wiki/Triplestore.

“Turtle (syntax).” Wikipedia, the Free Encyclopedia, July 13, 2013. http://en.wikipedia.org/w/index.php?title=Turtle_(syntax)&oldid=542183836.

Volz, Julius, Christian Bizer, Martin Gaedke, and Georgi Kobilarov. “Discovering and Maintaining Links on the Web of Data.” In The Semantic Web - ISWC 2009, edited by Abraham Bernstein, David R. Karger, Tom Heath, Lee Feigenbaum, Diana Maynard, Enrico Motta, and Krishnaprasad Thirunarayan, 5823:650–665. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009. http://www.springerlink.com/index/10.1007/978-3-642-04930-9_41.

Voss, Jon. “LODLAM State of Affairs.” Information Standards Quarterly 24, no. 2/3 (2012): 41. doi:10.3789/isqv24n2-3.2012.07.

W3C. “LinkedData.” Accessed August 4, 2013. http://www.w3.org/wiki/LinkedData.

“Welcome to Euclid.” Accessed September 4, 2013. http://www.euclid-project.eu/.

“Wiki.dbpedia.org : About.” Accessed November 3, 2013. http://dbpedia.org/About.

11 Scripts (done)

This section lists a set of computer source code implementing a simple linked data publishing system. Assuming the (Perl) developer has on hand a set of EAD finding aids and/or sets of MARC records, this system can:

  * transform EAD into RDF/XML
  * transform EAD into HTML
  * convert MARC into RDF/XML
  * convert MARC into HTML
  * support content negotiation against the RDF/XML and HTML
  * initialize a triple store
  * batch load RDF/XML into the triple store
  * search the triple store
  * dump the entire triple store as RDF/XML
  * support a SPARQL endpoint against the triple store

This publishing system is more than a toy and ought to be able to support the needs of many archives with small- to medium-sized collections.

11.a ead2rdf.pl - Perl script to make EAD files accessible via linked data (done)

```
# Eric Lease Morgan <eric_morgan@infomotions.com>
# December 6, 2013 - based on marc2linkedata.pl


# configure
use constant ROOT     => '/disk01/www/html/main/sandbox/liam';
use constant EAD      => ROOT . '/src/ead/';
use constant DATA     => ROOT . '/data/';
use constant PAGES    => ROOT . '/pages/';
use constant EAD2HTML => ROOT . '/etc/ead2html.xsl';
use constant EAD2RDF  => ROOT . '/etc/ead2rdf.xsl';
use constant SAXON    => 'java -jar /disk01/www/html/main/sandbox/liam/bin/saxon.jar -s:##SOURCE##
-xsl:##XSL## -o:##OUTPUT##';

# require
use strict;
use XML::XPath;
use XML::LibXML;
use XML::LibXSLT;

# initialize
my $saxon  = '';
my $xsl    = '';
my $parser = XML::LibXML->new;
my $xslt   = XML::LibXSLT->new;

# process each record in the EAD directory
my @files = glob EAD . "*.xml";
for ( 0 .. $#files ) {

    # re-initialize
    my $ead = $files[ $_ ];
    print "         EAD: $ead\n";
```

```
    # get the identifier
    my $xpath      = XML::XPath->new( filename => $ead );
    my $identifier = $xpath->findvalue( '/ead/eadheader/eadid' );
    $identifier    =~ s/[^\w ]//g;
    print "  identifier: $identifier\n";
    print "         URI: http://infomotions.com/sandbox/liam/id/$identifier\n";

    # re-initialize and sanity check
    my $output = PAGES . "$identifier.html";
    if ( ! -e $output or -s $output == 0 ) {

        # transform marcxml into html
        print "        HTML: $output\n";
        my $source     = $parser->parse_file( $ead )  or warn $!;
        my $style      = $parser->parse_file( EAD2HTML )   or warn $!;
        my $stylesheet = $xslt->parse_stylesheet( $style )  or warn $!;
        my $results    = $stylesheet->transform( $source )  or warn $!;
        my $html       = $stylesheet->output_string( $results );

        &save( $output, $html );

    }
    else { print "        HTML: skipping\n" }

    # re-initialize and sanity check
    my $output = DATA . "$identifier.rdf";
    if ( ! -e $output or -s $output == 0 ) {

        # create saxon command, and save rdf
        print "         RDF: $output\n";
        $saxon  =  SAXON;
        $xsl    =  EAD2RDF;
        $saxon  =~ s/##SOURCE##/$ead/e;
        $saxon  =~ s/##XSL##/$xsl/e;
        $saxon  =~ s/##OUTPUT##/$output/e;
        system $saxon;

    }
    else { print "         RDF: skipping\n" }

    # prettify
    print "\n";

}

# done
exit;


sub save {

    open F, ' > ' . shift or die $!;
    binmode( F, ':utf8' );
    print F shift;
    close F;
    return;
```

```
}
```

11.b marc2rdf.pl - Perl script to make MARC records accessible via linked data (done)

```
# Eric Lease Morgan <eric_morgan@infomotions.com>
# December 5, 2013 - first cut;


# configure
use constant ROOT      => '/disk01/www/html/main/sandbox/liam';
use constant MARC      => ROOT . '/src/marc/';
use constant DATA      => ROOT . '/data/';
use constant PAGES     => ROOT . '/pages/';
use constant MARC2HTML => ROOT . '/etc/MARC21slim2HTML.xsl';
use constant MARC2MODS => ROOT . '/etc/MARC21slim2MODS3.xsl';
use constant MODS2RDF  => ROOT . '/etc/mods2rdf.xsl';
use constant MAXINDEX  => 100;

# require
use IO::File;
use MARC::Batch;
use MARC::File::XML;
use strict;
use XML::LibXML;
use XML::LibXSLT;

# initialize
my $parser = XML::LibXML->new;
my $xslt   = XML::LibXSLT->new;

# process each record in the MARC directory
my @files = glob MARC . "*.marc";
for ( 0 .. $#files ) {

    # re-initialize
    my $marc = $files[ $_ ];
    my $handle = IO::File->new( $marc );
    binmode( STDOUT, ':utf8' );
    binmode( $handle, ':bytes' );
    my $batch  = MARC::Batch->new( 'USMARC', $handle );
    $batch->warnings_off;
    $batch->strict_off;
    my $index = 0;

    # process each record in the batch
    while ( my $record = $batch->next ) {

        # get marcxml
        my $marcxml =  $record->as_xml_record;
        my $_001    =  $record->field( '001' )->as_string;
        $_001       =~ s/_//;
        $_001       =~ s/ +//;
        $_001       =~ s/-+//;
        print "        marc: $marc\n";
        print "  identifier: $_001\n";
        print "         URI: http://infomotions.com/sandbox/liam/id/$_001\n";
```

```
        # re-initialize and sanity check
        my $output = PAGES . "$_001.html";
        if ( ! -e $output or -s $output == 0 ) {

            # transform marcxml into html
            print "        HTML: $output\n";
            my $source     = $parser->parse_string( $marcxml )  or warn $!;
            my $style      = $parser->parse_file( MARC2HTML )   or warn $!;
            my $stylesheet = $xslt->parse_stylesheet( $style )  or warn $!;
            my $results    = $stylesheet->transform( $source )  or warn $!;
            my $html       = $stylesheet->output_string( $results );

            &save( $output, $html );

        }
        else { print "        HTML: skipping\n" }

        # re-initialize and sanity check
        my $output = DATA . "$_001.rdf";
        if ( ! -e $output or -s $output == 0 ) {

            # transform marcxml into mods
            my $source     = $parser->parse_string( $marcxml )  or warn $!;
            my $style      = $parser->parse_file( MARC2MODS )   or warn $!;
            my $stylesheet = $xslt->parse_stylesheet( $style )  or warn $!;
            my $results    = $stylesheet->transform( $source )  or warn $!;
            my $mods       = $stylesheet->output_string( $results );

            # transform mods into rdf
            print "         RDF: $output\n";
            $source        = $parser->parse_string( $mods )     or warn $!;
            my $style      = $parser->parse_file( MODS2RDF )    or warn $!;
            my $stylesheet = $xslt->parse_stylesheet( $style )  or warn $!;
            my $results    = $stylesheet->transform( $source )  or warn $!;
            my $rdf        = $stylesheet->output_string( $results );

            &save( $output, $rdf );

        }
        else { print "         RDF: skipping\n" }

        # prettify
        print "\n";

        # increment and check
        $index++;
        last if ( $index > MAXINDEX )

    }

}

# done
exit;
```

```
sub save {

    open F, ' > ' . shift or die $!;
    binmode( F, ':utf8' );
    print F shift;
    close F;
    return;

}
```

11.c Dereference.pm - mod_perl module facilitate content negotiation (done)

```
# Dereference.pm - Redirect user-agents based on value of URI.

# Eric Lease Morgan <eric_morgan@infomotions.com>
# December 7, 2013 - first investigations; based on Apache2::Alex::Dereference
# January  7, 2014  - by default return HTML, not RDF


# configure
use constant PAGES => 'http://infomotions.com/sandbox/liam/pages/';
use constant DATA  => 'http://infomotions.com/sandbox/liam/data/';

# require
use Apache2::Const -compile => qw( OK );
use CGI;
use strict;

# main
sub handler {

    # initialize
    my $r   = shift;
    my $cgi = CGI->new;
    my $id  = substr( $r->uri, length $r->location );

    # wants html
    if ( $cgi->Accept( 'text/html' ) ) {

        print $cgi->header( -status => '303 See Other',
        -Location => PAGES . $id . '.html',
        -Vary     => 'Accept' ,
        "Content-Type" => 'text/html' )

    }

    # check for rdf
    elsif ( $cgi->Accept( 'application/rdf+xml' ) ) {

        print $cgi->header( -status => '303 See Other',
        -Location      => DATA . $id . '.rdf',
        -Vary          => 'Accept',
        "Content-Type" => 'application/rdf+xml' )

    }
```

```
    # give them html, anyway
    else {

        print $cgi->header( -status => '303 See Other',
        -Location => PAGES . $id . '.html',
        -Vary     => 'Accept' ,
        "Content-Type" => 'text/html' )

    }
    # done
    return Apache2::Const::OK;

}

1; # return true or die
```

11.d store-make.pl - a Perl script to simply initialize an RDF triple store (done)

```
# Eric Lease Morgan <eric_morgan@infomotions.com>
# December 14, 2013 - after wrestling with wilson for most of the day


# configure
use constant ETC => '/disk01/www/html/main/sandbox/liam/etc/';

# require
use strict;
use RDF::Redland;

# sanity check
my $db = $ARGV[ 0 ];
if ( ! $db ) {

    print "Usage: $0 <db>\n";
    exit;

}

# do the work; brain-dead
my $etc = ETC;
my $store = RDF::Redland::Storage->new( 'hashes', $db, "new='yes', hash-type='bdb', dir='$etc'" );
die "Unable to create store ($!)" unless $store;
my $model = RDF::Redland::Model->new( $store, '' );
die "Unable to create model ($!)" unless $model;

# "save"
$store = undef;
$model = undef;

# done
exit;
```

11.e store-add.pl - a Perl script to add items to an RDF triple store (done)

```
# Eric Lease Morgan <eric_morgan@infomotions.com>
```

```
# December 14, 2013 - after wrestling with wilson for most of the day


# configure
use constant ETC => '/disk01/www/html/main/sandbox/liam/etc/';

# require
use strict;
use RDF::Redland;

# sanity check #1 - command line arguments
my $db   = $ARGV[ 0 ];
my $file = $ARGV[ 1 ];
if ( ! $db or ! $file ) {

    print "Usage: $0 <db> <file>\n";
    exit;

}

# echo
warn "$file\n";

# sanity check #2 - store exists
die "Error: po2s file not found. Make a store?\n" if ( ! -e ETC . $db . '-po2s.db' );
die "Error: so2p file not found. Make a store?\n" if ( ! -e ETC . $db . '-so2p.db' );
die "Error: sp2o file not found. Make a store?\n" if ( ! -e ETC . $db . '-sp2o.db' );

# open the store
my $etc = ETC;
my $store = RDF::Redland::Storage->new( 'hashes', $db, "new='no', hash-type='bdb', dir='$etc'" );
die "Error: Unable to open store ($!)" unless $store;
my $model = RDF::Redland::Model->new( $store, '' );
die "Error: Unable to create model ($!)" unless $model;

# sanity check #3 - file exists
die "Error: $file not found.\n" if ( ! -e $file );

# parse a file and add it to the store
my $uri    = RDF::Redland::URI->new( "file:$file" );
my $parser = RDF::Redland::Parser->new( 'rdfxml', 'application/rdf+xml' );
die "Error: Failed to find parser ($!)\n" if ( ! $parser );
my $stream = $parser->parse_as_stream( $uri, $uri );
my $count  = 0;
while ( ! $stream->end ) {

    $model->add_statement( $stream->current );
    $count++;
    $stream->next;

}

# echo the result
#warn "Namespaces:\n";
#my %namespaces = $parser->namespaces_seen;
#while ( my ( $prefix, $uri ) = each %namespaces ) {
#
```

```
#   warn " prefix: $prefix\n";
#   warn '    uri: ' . $uri->as_string . "\n";
#   warn "\n";
#
#}
warn "Added $count statements\n";
warn "\n";

# "save"
$store = undef;
$model = undef;

# done
exit;
```

11.f store-search.pl - Perl script to query a triple store (done)

```
# Eric Lease Morgan <eric_morgan@infomotions.com>
# December 14, 2013 - after wrestling with wilson for most of the day


# configure
use constant ETC => '/disk01/www/html/main/sandbox/liam/etc/';
my %namespaces = (

  "crm"       => "http://erlangen-crm.org/current/",
  "dc"        => "http://purl.org/dc/elements/1.1/",
  "dcterms"   => "http://purl.org/dc/terms/",
  "event"     => "http://purl.org/NET/c4dm/event.owl#",
  "foaf"      => "http://xmlns.com/foaf/0.1/",
  "lode"      => "http://linkedevents.org/ontology/",
  "lvont"     => "http://lexvo.org/ontology#",
  "modsrdf"   => "http://simile.mit.edu/2006/01/ontologies/mods3#",
  "ore"       => "http://www.openarchives.org/ore/terms/",
  "owl"       => "http://www.w3.org/2002/07/owl#",
  "rdf"       => "http://www.w3.org/1999/02/22-rdf-syntax-ns#",
  "rdfs"      => "http://www.w3.org/2000/01/rdf-schema#",
  "role"      => "http://simile.mit.edu/2006/01/roles#",
  "skos"      => "http://www.w3.org/2004/02/skos/core#",
  "time"      => "http://www.w3.org/2006/time#",
  "timeline"  => "http://purl.org/NET/c4dm/timeline.owl#",
  "wgs84_pos" => "http://www.w3.org/2003/01/geo/wgs84_pos#"

);

# require
use strict;
use RDF::Redland;

# sanity check #1 - command line arguments
my $db    = $ARGV[ 0 ];
my $query = $ARGV[ 1 ];
if ( ! $db or ! $query ) {

    print "Usage: $0 <db> <query>\n";
    exit;
```

```
}

# sanity check #2 - store exists
die "Error: po2s file not found. Make a store?\n" if ( ! -e ETC . $db . '-po2s.db' );
die "Error: so2p file not found. Make a store?\n" if ( ! -e ETC . $db . '-so2p.db' );
die "Error: sp2o file not found. Make a store?\n" if ( ! -e ETC . $db . '-sp2o.db' );

# open the store
my $etc = ETC;
my $store = RDF::Redland::Storage->new( 'hashes', $db, "new='no', hash-type='bdb', dir='$etc'" );
die "Error: Unable to open store ($!)" unless $store;
my $model = RDF::Redland::Model->new( $store, '' );
die "Error: Unable to create model ($!)" unless $model;

# search
#my $sparql  = RDF::Redland::Query->new( "CONSTRUCT { ?a ?b ?c } WHERE { ?a ?b ?c }", undef, undef,
"sparql" );
my $sparql = RDF::Redland::Query->new( "PREFIX modsrdf:
<http://simile.mit.edu/2006/01/ontologies/mods3#>\nSELECT ?a ?b ?c WHERE {  ?a  modsrdf:$query ?c }",
undef, undef, 'sparql' );
my $results = $model->query_execute( $sparql );
print $results->to_string;

# done
exit;


11.g  sparql.pl - a Perl-based, brain-dead, half-baked SPARQL endpoint (done)

# Eric Lease Morgan <eric_morgan@infomotions.com>
# December 15, 2013 - first investigations


# require
use CGI;
use CGI::Carp qw( fatalsToBrowser );
use RDF::Redland;
use strict;

# initialize
my $cgi   = CGI->new;
my $query = $cgi->param( 'query' );

if ( ! $query ) {

    print $cgi->header;
    print &home;

}

else {

    # open the store for business
    my $store = RDF::Redland::Storage->new( 'hashes', 'store', "new='no', hash-type='bdb',
dir='/disk01/www/html/main/sandbox/liam/etc'" );
    my $model = RDF::Redland::Model->new( $store, '' );
```

```
    # search
    my $results = $model->query_execute( RDF::Redland::Query->new( $query, undef, undef, 'sparql' ) );

    # return the results
    print $cgi->header( -type => 'application/xml' );
    print $results->to_string;

}

# done
exit;


sub home {

    # return a home page
    return <<EOF
<html>
<head>
<title>LiAM SPARQL Endpoint</title>
</head>
<body style='margin: 7%'>
<h1>LiAM SPARQL Endpoint</h1>
<p>This is a brain-dead and half-baked SPARQL endpoint to a subset of LiAM linked data. Enter a query, but
there is the disclaimer. Errors will probably happen because of SPARQL syntax errors. Remember, the
interface is brain-dead. Your milage <em>will</em> vary.</p>
<form method='GET' action='./'>
<textarea style='font-size: large' rows='5' cols='65' name='query' />
SELECT ?p ?o
WHERE { &lt;http://infomotions.com/sandbox/liam/id/mum432&gt; ?p ?o }
ORDER BY ?p
</textarea><br />
<input type='submit' value='Search' />
</form>
<p>Sample queries:</p>
<ul>
    <li>All the classes in the triple store - <code><a
href="http://infomotions.com/sandbox/liam/sparql/?query=SELECT+DISTINCT+%3Fo+WHERE+%7B+%3Fs+a+%3Fo+%7D+
ORDER+BY+%3Fo" target="_blank">SELECT DISTINCT ?o WHERE { ?s a ?o } ORDER BY ?o</a></code></li>
    <li>All the properties in the triple store - <code><a
href="http://infomotions.com/sandbox/liam/sparql/?query=SELECT+DISTINCT+%3Fp+WHERE+%7B+%3Fs+%3Fp+%3Fo+%7D+
ORDER+BY+%3Fp" target="_blank">SELECT DISTINCT ?p WHERE { ?s ?p ?o } ORDER BY ?p</a></code></li>
    <li>All the things things that are archival finding aids - <code><a
href="http://infomotions.com/sandbox/liam/sparql/?query=SELECT+%3Fs+WHERE+%7B+%3Fs+a+%3Chttp%3A%2F%2Fdata.
archiveshub.ac.uk%2Fdef%2FFindingAid%3E+%7D" target="_blank">SELECT ?s WHERE { ?s a
&lt;http://data.archiveshub.ac.uk/def/FindingAid&gt; }</a></code></li>
    <li>Everything about a specific actionable URI (finding aid) - <code><a
href="http://infomotions.com/sandbox/liam/sparql/?query=SELECT+%3Fp+%3Fo+WHERE+%7B+%3Chttp%3A%2F%
2Finfomotions.com%2Fsandbox%2Fliam%2Fid%2Fmum432%3E+%3Fp+%3Fo+%7D+ORDER+BY+%3Fp" target="_blank">SELECT ?p
?o WHERE { &lt;http://infomotions.com/sandbox/liam/id/mum432&gt; ?p ?o } ORDER BY ?p</a></code></li>
    <li>Ten things that are MARC records - <code><a
href="http://infomotions.com/sandbox/liam/sparql/?query=SELECT+%3Fs+WHERE+%7B+%3Fs+a+%3Chttp%3A%2F%
2Fsimile.mit.edu%2F2006%2F01%2Fontologies%2Fmods3%23Record%3E+%7D+LIMIT+10" target="_blank">SELECT ?s
WHERE { ?s a &lt;http://simile.mit.edu/2006/01/ontologies/mods3#Record&gt; } LIMIT 10</a></code></li>
    <li>Everything about a specific actionable URI - <code><a
href="http://infomotions.com/sandbox/liam/sparql/?query=SELECT+%3Fp+%3Fo+WHERE+%7B+%3Chttp%3A%2F%
```

```
2Finfomotions.com%2Fsandbox%2Fliam%2Fid%2Fshumarc681792%3E+%3Fp+%3Fo+%7D+ORDER+BY+%3Fp"
target="_blank">SELECT ?p ?o WHERE { &lt;http://infomotions.com/sandbox/liam/id/shumarc681792&gt; ?p ?o }
ORDER BY ?p</a></code></li>
    <li>One hundred things with titles - <code><a
href="http://infomotions.com/sandbox/liam/sparql/?query=SELECT+%3Fs+%3Fo+WHERE+%7B+%3Fs+%3Chttp%3A%2F%
2Fpurl.org%2Fdc%2Fterms%2Ftitle%3E+%3Fo+%7D+ORDER+BY+%3Fo+LIMIT+100" target="_blank">SELECT ?s ?o WHERE {
?s &lt;http://purl.org/dc/terms/title&gt; ?o } ORDER BY ?o LIMIT 100</a></code></li>
    <li>One hundred things with creators - <code><a
href="http://infomotions.com/sandbox/liam/sparql/?query=SELECT+%3Fs+%3Fo+WHERE+%7B+%3Fs+%3Chttp%3A%2F%
2Fsimile.mit.edu%2F2006%2F01%2Froles%23creator%3E+%3Fo+%7D+ORDER+BY+%3Fo+LIMIT+100" target="_blank">SELECT
?s ?o WHERE { ?s &lt;http://simile.mit.edu/2006/01/roles#creator&gt; ?o } ORDER BY ?o LIMIT
100</a></code></li>
    <li>One hundred things with names - <code><a
href="http://infomotions.com/sandbox/liam/sparql/?query=SELECT+%3Fs+%3Fo+WHERE+%7B+%3Fs+%3Chttp%3A%2F%
2Fxmlns.com%2Ffoaf%2F0.1%2Fname%3E+%3Fo+%7D+ORDER+BY+%3Fo+LIMIT+100" target="_blank">SELECT ?s ?o WHERE {
?s &lt;http://xmlns.com/foaf/0.1/name&gt; ?o } ORDER BY ?o LIMIT 100</a></code></li>
    <li>One hundred things with notes - <code><a
href="http://infomotions.com/sandbox/liam/sparql/?query=SELECT+%3Fs+%3Fo+WHERE+%7B+%3Fs+%3Chttp%3A%2F%
2Fdata.archiveshub.ac.uk%2Fdef%2Fnote%3E+%3Fo+%7D+ORDER+BY+%3Fo+LIMIT+100" target="_blank">SELECT ?s ?o
WHERE { ?s &lt;http://data.archiveshub.ac.uk/def/note&gt; ?o } ORDER BY ?o LIMIT 100</a></code></li>
</ul>
<p>For more information about SPARQL, see:</p>
<ol>
    <li><a href="http://www.w3.org/TR/rdf-sparql-query/" target="_blank">SPARQL Query Language for RDF</a>
from the W3C</li>
    <li><a href="http://en.wikipedia.org/wiki/SPARQL" target="_blank">SPARQL</a> from Wikipedia</li>
</ol>
<p>Source code -- <a href="http://infomotions.com/sandbox/liam/bin/sparql.pl">sparql.pl</a> -- is
available online.</p>
<hr />
<p>
<a href="mailto:eric_morgan\@infomotions.com">Eric Lease Morgan
&lt;eric_morgan\@infomotions.com&gt;</a><br />
March 5, 2014
</p>
</body>
</html>
EOF
}
```

11.h store-dump.pl - Perl script to output the content of store as RDF/XML (done)

```
# Eric Lease Morgan <eric_morgan@infomotions.com>
# December 14, 2013 - after wrestling with wilson for most of the day


# configure
use constant ETC => '/disk01/www/html/main/sandbox/liam/etc/';

# require
use strict;
use RDF::Redland;

# sanity check #1 - command line arguments
my $db  = $ARGV[ 0 ];
if ( ! $db ) {
```

```
    print "Usage: $0 <db>\n";
    exit;

}

# sanity check #2 - store exists
die "Error: po2s file not found. Make a store?\n" if ( ! -e ETC . $db . '-po2s.db' );
die "Error: so2p file not found. Make a store?\n" if ( ! -e ETC . $db . '-so2p.db' );
die "Error: sp2o file not found. Make a store?\n" if ( ! -e ETC . $db . '-sp2o.db' );

# open the store
my $etc = ETC;
my $store = RDF::Redland::Storage->new( 'hashes', $db, "new='no', hash-type='bdb', dir='$etc'" );
die "Error: Unable to open store ($!)" unless $store;
my $model = RDF::Redland::Model->new( $store, '' );
die "Error: Unable to create model ($!)" unless $model;

# do the work
my $serializer = RDF::Redland::Serializer->new;
print $serializer->serialize_model_to_string( RDF::Redland::URI->new, $model );

# done
exit;
```

11 A question from a library school student (done)

In January of 2014 I received the following email message from a library school student. The questions they asked were apropos to the Guide, so I thought I'd include my response here, but the names have been changed to protect the innocent.

From: Eric Lease Morgan <emorgan@nd.edu>
Subject: Re: RDF ontologies discussion on Code4Lib Listserv
Date: January 21, 2014 at 9:36:36 PM EST

> I'm writing you to ask you about your thoughts on implementing
> these kinds of RDF descriptions for institutional collections.
> Have you worked on a project that incorporated LD technologies
> like these descriptions? What was that experience like? Also,
> what kind of strategies have you used to implement these
> strategies, for instance, was considerable buy-in from your
> organization necessary, or were you able to spearhead it
> relatively solo? In essence, what would it "cost" to really do
> this?
>
> I apologize for the mass of questions, especially over e-mail. My
> only experience with ontology work has been theoretical, and I
> haven't met any professionals in the field yet who have actually
> used it. When I talk to my mentors about it, they are aware of it
> from an academic standpoint but are wary of it due these
> questions of cost and resource allocation, or confusion that it
> doesn't provide anything new for users. My final project was to
> build an ontology to describe some sort of resource and I settled
> on building a vocabulary to describe digital facsimiles and their
> physical artifacts, but I have yet to actually implement or use
> any of it. Nor have I had a chance yet to really use any
> preexisting vocabularies. So I've found myself in a slightly
> frustrating position where I've studied this from an academic
> perspective and seek to incorporate it in my GLAM work, but I
> lack the hands-on opportunity to play around with it.
>
> --
> MLIS Candidate

Dear MLS Candidate, no problem, really, but I don't know how much help I will really be.

The whole RDF / Semantic Web thing started more than ten years ago. The idea was to expose RDF/XML, allow robots to crawl these files, amass the data, and discover new knowledge – relationships – underneath. Many in the library profession thought this was science fiction and/or the sure pathway to professional obsolescence. Needless to say, it didn't get very far. A few years ago the idea of linked data was articulated, and it a nutshell it outlined how to make various flavors of serialized RDF available via an HTTP technique called content negotiation. This was when things like Turtle, N3, the idea of triple stores, maybe SPARQL, and other things came to fruition. This time the idea of linked data was more real and got a bit more traction, but it is still not main stream.

I have very little experience putting the idea of RDF and linked data into practice. A long time ago I created RDF versions of my Alex Catalogue and implemented a content negotiation tool against it. The Catalogue was not a part of any institution other than myself. When I saw the call for the LiAM Guidebook I applied and got the "job" because of my Alex Catalogue experiences as well as some experience with a thing colloquially called The Catholic Portal which contains content from EAD files.

I knew this previously, but linked data is all about URIs and ontologies. Minting URIs is not difficult,

but instead of rolling your own, it is better to use the URIs of others, such as the URIs in DBpedia, GeoNames, VIAF, etc. The ontologies used to create relationships between the URIs are difficult to identify, articulate, and/or use consistently. They are manifestations of human language, and human language is ambiguous. Trying to implement the nuances of human language in computer "sentences" called RDF triples is only an approximation at best. I sometimes wonder if the whole thing can really come to fruition. I look at OAI-PMH. It had the same goals, but it was finally called not a success because it was too difficult to implement. The Semantic Web may follow suit.

That said, it is not too difficult to make the metadata of just about any library or archive available as linked data. The technology is inexpensive and already there. The implementation will not necessarily implement best practices, but it will not expose incorrect nor invalid data, just data that is not the best. Assuming the library has MARC or EAD files, it is possible to use XSL to transform the metadata into RDF/XML. HTML and RDF/XML versions of the metadata can then be saved on an HTTP file system and disseminated either to humans or robots through content negotiation. Once a library or archive gets this far they can then either improve their MARC or EAD files to include more URIs, they can improve their XSLT to take better advantage of shared ontologies, and/or they can dump MARC and EAD all together and learn to expose linked data directly from (relational) databases. It is an iterative process which is never completed.

Nothing new to users? Ah, that is the rub and a sticking point with the linked data / Semantic Web thing. It is a sort of chicken & egg problem. "Show me the cool application that can be created if I expose my metadata as linked data", say some people. On the other hand, "I can not create the cool application until there is a critical mass of available content." Despite this issue, things are happening for readers, namely mash-ups. (I don't like the word "users".) Do a search in Facebook for the Athens. You will get a cool looking Web page describing Athens, who has been there, etc. This was created by assembling metadata from a host of different places (all puns intended), and one of those places were linked data repositories. Do a search in Google for the same thing. Instead of just bringing back a list of links, Google presents you with real content, again, amassed through various APIs including linked data. Visit VIAF and search for a well-known author. Navigate the result an you will maybe end up at WorldCat identities where all sorts of interesting information about an author, who they wrote with, what they wrote, and where is displayed. All of this is rooted in linked data and Web Services computing techniques. This is where the benefit comes. Library and archival metadata will become part of these mash-up – called "named graphs" – driving readers to library and archival websites. Linked data can become part of Wikipedia. It can be used to enrich descriptions of people in authority lists, gazetteers, etc.

What is the cost? Good question. Time is the biggest expense. If a person knows what they are doing, then a robust set of linked data could be exposed in less than a month, but in order to get that far many people need to contribute. Systems types to get the data out of content management systems as well as set up HTTP servers. Programmers will be needed to do the transformations. Catalogers will be needed to assist in the interpretation of AACR2 cataloging practices, etc. It will take a village to do the work, and that doesn't even count convincing people this is a good idea.

Your frustration is not uncommon. Often times if there is not a real world problem to solve, learning anything new when it comes to computers is difficult. I took BASIC computer programming three times before anything sunk in, and it only sunk in when I needed to calculate how much money I was earning as a taxi driver.

Try implementing one of your passions. Do you collect anything? Baseball cards? Flowers? Books? Records? Music? Art? Is there something in your employer's special collections of interest to you? Find something of interest to you. For simplicity's sake, use a database to describe each item in the collection making sure each record in our database includes a unique key field. Identify one or more ontologies (others as well as rolling your own) whose properties match closely the names of your fields in your database. Write a program against your database to create static HTML pages. Put the pages on the Web. Write a program against your database to create static RDF/XML (or some other RDF serialization). Put the pages on the Web. Implement a content negotiation script that takes the keys to your database's fields as input and redirects HTTP user agents to either the HTML or RDF. Submit the root of your linked data implementation to Datahub.io. Ta da! You have successfully implemented linked data and learned a whole lot along the way.

Once you get that far you can take what you have learned and apply it in a bigger and better way for a larger set of data.

On one hand the process is not difficult. It is a matter of repurposing the already existing skills of people who work in cultural heritage institutions. On the other hand, change in the ways things are done is difficult (but not as difficult in the what of what is done). The change is difficult to balance existing priorities.

Exposing library and archival content as linked data represents a different working style, but the end result is the same – making the content of our collections available for use and understanding.

HTH.

—

Eric Morgan

Out takes

The Semantic Web is about encoding data, information, and knowledge in computer-readable fashions, making these encodings accessible on the World Wide Web, allowing computers to crawl the encodings, and finally, employing reasoning engines against them for the purpose of discovering and creating new knowledge. The following section are a primer to the principles and practices of linked data.

linked open Data

For all intents and contexts, linked data and linked open data are synonymous, but the subtle difference does need to be discussed. Linked open data is a qualification of linked data. Linked open data comes with an explicit license agreement denoting how the accessible data can be "freely" used. In this case, the words "free" and "open" are analogous to free "open source software". Just as open source software is available for use and re-use, linked open data is free for use and re-use. Attribution needs to be made. The data can be freely used. While copyrights may still be in place when it comes to linked open data, the copyrights allow for the use, re-use, and re-distribution. The intent of linked open data is to use the content in ways that it canb e used in many ways for many purposes. While the distinction between linked data and linked open data are may be large in the eyes of some people, for simplicities sake, the phrase linked data is synonymous with linked open data, even though some feel the distinction needs to be delineated to a greater degree.

Linked data is a standardized process for publishing and disseminating information via the Web. It represents the current "how" behind the ideas the Semantic Web.

Increasingly you will hear of of linked data being qualified as "linked open data". The "open" qualifier alludes to the important distinctions between truly free data/information and licensed data/information coming with strings attached. Truly "open" linked data comes with no financial restrictions or restrictions on use, but there may very well be attribution requirements.