Publishing Bibliographic Records on the Web of Data: Opportunities for the BnF (French National Library)

Agnès Simon[1], Romain Wenz[1], Vincent Michel[2], Adrien Di Mascio[2]

[1] Bibliothèque nationale de France, Paris, France
agnes.simon@bnf.fr, romain.wenz@bnf.fr
[2] Logilab, Paris, France
adrien.dimascio@logilab.fr, vincent.michel@logilab.fr

Abstract. Linked open data tools have been implemented through data.bnf.fr, a project which aims at making the BnF data more useful on the Web. data.bnf.fr gathers data automatically from different databases on pages about authors, works and themes. Online since July 2011, it is still under development and has feedbacks from several users, already.
First the article will present the issues linked to our data and stress the importance of useful links and of persistency for archival purposes. We will discuss our solution and methodology, showing their strengths and weaknesses, to create new services for the library. An insight on the ontology and vocabularies will be given, with a "business" view of the interaction between rich RDF ontologies and light HTML embedded data such as schema.org. The broader question of Libraries on the Semantic Web will be addressed so as to help specify similar projects.

Keywords: Libraries, Open data, Culture, Project management, Entity linking, Relational database, CubicWeb, Encoded Archival Description, Marc formats, Open Archive Initiative, Text Encoding Initiative.

1 Introduction

The BnF (French national library) sees Semantic Web technologies as an opportunity to weave its data into the Web and to bring structure and reliability to existing information. The BnF is one of the most important heritage institutions in France, with a history going back to the 14th century and millions of documents, including a large variety of hand-written, printed and digital material, through millions of bibliographic records. Linked Open Data tools have been implemented through data.bnf.fr, a project which aims at making the BnF data more useful on the Web.

data.bnf.fr publishes data automatically merged from different in-house databases describing authors, works and themes. These concepts are given persistent URIs, as they are the nodal points to our resources and services. We provide **different views of the same information**: HTML and PDF views for humans and raw data in RDF and JSON for machines. This data is freely reusable under an **Open License**. The site, powered by the **open source platform** CubicWeb, queries a **relational database** to generate both HTML and RDF data. Available online since July 2011, this service is under continuous development with several releases per year. After having gathered feedback from the public and users, we are now in a position to report on this use of Semantic Web technologies.

We want to show how we transform a mass of bibliographical data from different databases, to display structured and reliable data in liked data: what were the difficulties and the solutions? What are the impacts in terms of services for libraries and for the wider community of the Web?

2 Context and Goals

2.1 Strength and Weakness of our Data

The BnF (French national Library) took a first step towards the Web with the digital library Gallica (`http://gallica.bnf.fr`)[1], which offers over 2 million documents like books, reviews, images, objects, and scores. Yet when it comes to our data, it sometimes remains hard to find it especially as users have new expectations and habits on the Web. They need to reach digital collections and references to physical documents through simple keyword searches via search engines and "following their nose" from one link to another. As we cannot ask an always broader audience to become familiar with our various catalogues, we have to help them getting oriented in this mass of data. Indeed the BnF holds millions of documents and descriptive data, especially:

- from the "Catalogue général" (`http://catalogue.bnf.fr/`), which is the main catalogue with about 11 millions of bibliographical data including all the French Legal Deposit,
- from the Archives and Manuscripts database (`http://archivesetmanuscrits.bnf.fr/`), with around 150 000 records,
- from the authority files, with more than 2 million authority records on persons, organizations, works or subjects, and structured repositories, such as the list of roles (`http://data.bnf.fr/vocabulary/roles`) that have a value on the Web.

For machines this data is hard to handle: it is hidden in the deep Web, unstructured, and stored in relational databases. The information has originally been produced to manage our collections, before Web standards even existed. Besides, descriptions are maintained in disparate BnF catalogues, reflecting the methods and technologies used for their descriptions. They have been produced in different formats, according to the type of document that is described. For instance, a collection of archives and manuscripts needs a hierarchical structure, to describe documents together, as they were produced and received during the activities of a person. Therefore archives are described in EAD-XML formats, adapted to a hierarchical "tree structure", whereas books and reviews from the main catalogue are described in a MARC format, created in the 1960's for the librarian community and displaying a flat series of records [2][3][4].

Nevertheless libraries have been playing a major role in normalizing data and respecting cataloguing codes, norms and formats. A first step has been taken by adopting XML (TEI [5], EAD-XML [6], Dublin Core [7] for instance) formats to create or exchange data. We also have been using permanent and reliable "ARK" identifiers [8], to identify catalogue records, archival resources, digital objects from Gallica, and authority records, but also for quoting these resources, with a common "resolver". For instance, the digital object
`http://gallica.bnf.fr/ark:/12148/bpt6k134521m` is also accessible with

the persistent link http://ark.bnf.fr/ark:/12148/bpt6k134521m. Furthermore these identifiers have been used inside the library to link our bibliographical records from the main catalogue and our archival finding aids in XML to the authority records. Some of these links are already "typed". For instance, the link between a book and its author or contributor is usually specified by a role code, listed and controlled in our repositories. Charles Baudelaire is the translator of this edition of Dix contes by Edgar Poe: the record http://catalogue.bnf.fr/ark:/12148/cb311263053 is linked to the author with role "translator", expressed with the code 0680.

Thus the work on standards and identifiers has made it possible for libraries to become part of the semantic Web (http://www.ifla.org/about-swsig). To do so, library data has to become both linked, and open.

2.2 Business Issues: Libraries in Linked Open Data

This project takes part of the international experimentations of "Linked Open Data" (LOD), that have popped up among national libraries. The Library of Congress, the Deutsche Nationalbibliothek or the British Library display all their bibliographical records in RDF. Libraries across the world show interest on these topics, in a passionate and sometimes controversial way [9], recommending to identify sets of data as "possible candidates for early exposure as Linked Data and foster a discussion about Open Data". It is also an incentive for others to use this material and to give access to culture. That way, the BnF is taking part of the "Open data" movement, to give access to the information to the broader public, by using the most recent technologies. That is why we chose the "Open License" that allows any commercial use under the condition of quoting the source "Bibliothèque nationale de France".

Yet the BnF had specific goals and issues. First we had to deal with different databases, to link metadata of paper documents with its digitized version, or to gather archives with published documents. We had to transform data from non-interoperable databases into structured and exchangeable data compatible with Semantic Web standards. The workflow was to take our data as it is, with its faults and assets, to keep the global data producing process and our existing catalogues. We chose to keep separate the archival base from the bibliographical base, "upstream", and to display data on the Web with common vocabularies, "downstream".

Secondly, data.bnf.fr also builds *pages for humans*, whereas most big libraries display their bibliographic data in triple stores as another kind of bibliographic product for libraries. The quality of the data being irregular as a result of the long history of our catalogue, this data is displayed gradually on the Web.

Finally, as the main purpose of the library is to give access to documents for patrons, the HTML publication had to be coherent with the RDF publication, the data in RDF being just a different view from the same data that is in the HTML page. The URIs displayed in the RDF give actual links to relevant and existing information, which makes the issue on identifiers so important. Besides the RDF and the HTML data model are similar. We chose basic concepts that are relevant for creating a Web page: authors, works, and subjects. It happened to be an opportunity to implement the FRBR (fundamental requirements for bibliographic records) model [10] which is mainly based on three entities (author, work, and subject) and on the difference between the work, as an intellectual creation, the versions of this work like a translation or an illustrated edition ("expression") and the publication of this creation ("manifestation"). For instance, in data.bnf.fr, you find pages about the work Tales of the grotesque and arabesque (`http://data.bnf.fr/11943795/edgar_allan_poe_histoires_extraordinaires/`) where we gather the different editions of this work. On the page about the author *"Charles Baudelaire"* (`http://data.bnf.fr/11890582/charles_baudelaire/`), we gather links to his works such as `"Le Spleen de Paris"` and to his specific contributions on publications as translator, illustrator, or dedicator. . . , at the good level of the model. Considering our needs and issues and from a business side, we finally chose to rely on CubicWeb on the following grounds: it is an open source software; it can extract data from different databases, match them and gather them from different databases; it can publish the same information in different views (Web pages for humans, as well as structured data for computers).

2.3 Technical Considerations

Since all the workflow would rely on documenting provenance and merging information, the use of a triple store was not obvious, and did not appear as the only possible option. Alignments had to be made between various sources: several datasets had to be matched and linked. But most of them were already exposed on the "Linked Open Data cloud", with reliable and efficient links. Whenever it was possible, we used existing matching « Hubs » like the Virtual International Authority File (VIAF), now available in Open Data or DBpedia, as go-between to link to other sets, so as to get the best results with the available time and effort. VIAF is an international project to federate the different authority files from many national libraries. We plan to keep using DBpedia in the future to link to other datasets, that are linked to it. Consuming information which is available with an open license is one of the key issues for using existing matchings. We are also creating new matchings, for instance for geographic entities, which imply making an alignment between our geographic subject headings, Geonames, and our "Rameau" subject headings which is used in the library records.

The BnF chose CubicWeb among other software solutions (including triple Stores), because it had good references, a cost-efficient development, and an ability to publish data with Semantic Web standards (RDF, SPARQL, HTML5, CSS3, Responsive Design) and to cope with our data in several formats. It appeared as one of the most advanced open source Python frameworks for data management. In the next section, we will explain this choice by detailing the advantages and drawbacks of this approach, which reflects the common questions that appear when Semantic Web projects enter a production environment and have to become "business as usual".

3 CubicWeb in a Nutshell

CubicWeb is a Semantic Web application framework, licensed under the LGPL. It relies on different widely-used and well established technologies (see Fig.1 for the global architecture of CubicWeb):

- SQL frameworks for the databases (e.g. sqlite, MySql, PostgreSql),

- Python for the core code and the web server,
- Javascript for the client-side logic.

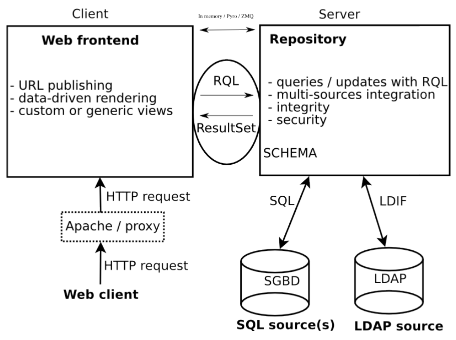

Fig. 1. Overview of CubicWeb architecture

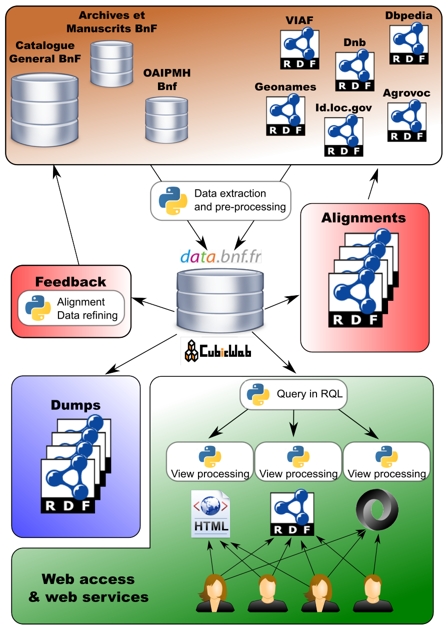

Fig. 2. Global architecture of Data.bnf.fr

3.1 Schema, Views and Entities

CubicWeb applications are structured in three main parts:

- a *schema*, i.e. data model,
- some *views*, i.e information publishing (HTML, PDF, RDF, etc.),
- some *entity classes*, i.e. business logic.

The *schema* defines the data model in terms of attributes/relations/constraints. It is written in Python, and makes the description of the data very simple. Here is an example of the data.bnf.fr schema for an Author (cf. Snippet 1).

With CubicWeb, the result of queries is presented by applying functions named "views". A top-level view can generate a Web page, but also generate a PDF or a JSON file. This is a key distinction when comparing CubicWeb with Web frameworks that are centered on Web pages and not on data. Frameworks centered on Web pages use templates to introduce dynamic content, with a template language that usually becomes cumbersome if one needs more than loops and tests. With CubicWeb, a Web page is a call to the top-level view that calls other views, each of these views call other views, down to the basic text properties of objects. Using templates is possible, but requires having a piece of template for each function/view. We tried to use templates, but it proved more efficient and readable not to split things into two and to directly emit HTML, XML, text or binary data directly to the output stream. *Views* define the different (fine-grained) ways of displaying data. These views could write (chunk of) HTML pages, but also RDF, CSV, JSON, XML, PDF, etc... (cf. Snippet 2) Each output representation therefore uses the same data. There are no static dumps of data, and every visualization are always up to date. These views may be also used to visualize the same result set in different ways based on the actual context: a result set of works will be displayed differently if we are in the page of their author, or if we are on the page of listing of all the works in the database.

Entity classes define business logic. Some logic could be added to entities, by adding python functions building attributes or relations.

Snippet 1 This defines a *Person*, with a *gender* and a *birthplace*, a link to a *PersonDefinition* called *preferred_form*, and multiple *other_forms*. The *PersonDefinition* is an entity with two attributes *name* and *surname*.

```
class Person(EntityType):
    preferred_form = SubjectRelation('PersonDefinition', cardinality='1?')
    other_forms = SubjectRelation('PersonDefinition', cardinality='1*')
    gender = String(vocabulary=(_('M'), _('F')))
    birthplace = String(maxsize=128)
    # ...

class PersonDefinition(EntityType):
    surname = String(maxsize=256)
    firstname = String(maxsize=128)
    # ...
```

3.2 Relation Query Language

CubicWeb uses a homemade query language, called RQL, which is similar to W3C's query language SPARQL ahd that has been developed since 2001. This language is closely related to the underlying data model, and is used to query the data. It is based on syntax analysis of the query, and can infer information such as entity type from the query. Development of RQL started in 2001. When the normalization process that led to SPARQL started at W3C, Logilab (the company behind CubicWeb) had not enough manpower to participate. Therefore, RQL has been developed in parallel to SPARQL. The two languages share common goals and focus on relationships. Yet there are a few interesting differences between them:

- RQL syntax is often simpler with less punctuation signs and braces,
- RQL has always allowed INSERT, SET and DELETE operations,
- RQL is easily extended with functions, for example, using PostGIS (GIS system based on Postgresql) one can use in RQL the INTERSECTS function defined with the PL/PGSQL procedural language,

– Experience shows that RQL is quickly adopted by power-users, especially with a directive usable in restructured text wiki-like pages `:rql:`<some_query>:<vid>``

Here are a few RQL queries that are used in data.bnf.fr :

– `Any X WHERE X notice_id 12345678` will return any object in the database that has this notice id, whether it be a work, an author, etc.
– `Any R WHERE X is Rameau, X eid 1234, X broader_concept R` will return all Rameau objects that are broader concepts of the Rameau with internal id 1234,
– `Any X WHERE X is Person, X preferred_form P, P surname ILIKE "A%"` will return all persons whose preferred form's surname starts with 'A' or 'a',

This language is the only way to talk to the database, it supports the four LMD basic operations *Any* (read), *INSERT* (create), *SET* (update), *DELETE* (delete), and also subqueries, orderby, aggregation, functions.

Snippet 2 This view generates the *Other resources* part of an author or work page

```
class OtherRessourcesView(EntityView):
    __regid__ =  'other-ressources'
    __select__ = EntityView.__select__ & is_instance(*DU_ETYPES)

    def cell_call(self, row, col):
        # get the current entity (author or work)
        entity = self.cw_rset.get_entity(row, col)
        if entity.has_other_resources:
            self.w(u'<div class="section" id="other-ressources">')
            self.w(u'<h2>%s</h2>' % self._cw._("other ressources").capitalize())
            # display virtual exhibitions
            self.w(entity.view('virtual-exhibitions'))
            # display BnF's aligned bookmarks
            self.w(entity.view('bnf-bookmarks'))
            # If an author is aligned on dbpedia, display its short abstract
            self.w(entity.view('dbpedia'))
            self.w(u'</div>')
```

3.3 Security

The permission definition is an integral part of the data model definition in a CubicWeb application (cf. Snippet 3). More specifically, the permission model is very simple in data.bnf.fr since nearly everything is readable by anyone. A simple workflow is attached to each kind of entity in the internal data model (e.g. *Person*, *Work*, etc.). Those entities can be temporarily unpublished by an agent using the administration Web interface (which is actually the very same Web application as the official Web site).

With this permissions, the query `Any X WHERE X is Person` will be executed as is if the user is in the *managers* or *users* group but will otherwise be transformed into `Any X WHERE X is Person, X visible TRUE`

This is a real time saver since, as a programmer, you actually don't have to worry about permissions when writing queries, the repository will never return something that the connected user is not allowed to read. The same logic applies for *add*, *update* or *delete* queries.

Snippet 3 Permissions used on main entities: to be able to read an entity, the user issuing the query must either be in one the *managers* or *users* groups. Otherwise, the CubicWeb repository will inject the RQL expression `X visible TRUE` in the original query to make sure that only entities matching this condition will be returned

```
__permissions__ = {
    'read': ('managers', 'users', ERQLExpression('X visible TRUE')),
    'update': ('managers',
               ERQLExpression('U in_group G, G name "users", X in_state S, '
                              'S name IN "temporary-unpublished"')),
    'delete': ('managers',),
    'add': ('managers',)}
```

4 Semantic Point of View

The heart of a CubicWeb application is the data model. Once this data model is defined, the framework is able to generate a database and a Web application instance to add, store, browse and query data fulfilling this data model (see Fig.2 for the global architecture of data.bnf.fr).

4.1 Using Relational Databases

CubicWeb applications have been deployed, used and maintained for 10 years, a time period where quite a few Semantic Web standards were still emerging. For this reason, we decided to stick to well established standards and used SQL relational databases: the knowledge base is huge, every system administration team knows how to deal with them, how to optimize them, how to replicate them, etc. Major Websites can be built upon a triple store, but the SQL relation databases have, in our opinion, a bit more feedbacks from industrial use and make it easier to interact with existing teams who work on relational databases (library catalogues) inside the library. From the library's point of view, it is also an opportunity to keep the producing formats (EAD, MARC. . .) and workflows, as they are. Furthermore, while RDF is the *de facto* standard in the Semantic Web world for data input/output, Semantic Web applications don't need to rely on a triple-store for internal data management.

In our case, we need to absorb different kind of data, structured or not (Marc XML data, RDF-NT, RDF-XML, CSV files or dumps of relational databases), and therefore using SQL database(s) as a pivot for melting all these data may be interesting. Where triplestores are the natural choice for storing and querying RDF data, we need in our case to serve thousands of daily views, for more than 200.000 web pages, in a rich variety of formats (RDF/JSON/CSV/HTML). This implies a strong structuration and control of the data we put in, and a better integration in a complete Web application. Relational databases and the underlying relational algebra field have been studied for years and have reached both theoretical and practical maturity needed for such applications. Furthermore, with open source SQL backends, we benefit from a huge knowledge base, large communities, and a lot of cookbooks for deployments, optimizations, debugging, etc.

One Model Definition, Several Ontologies Used in Published Data. Using a SQL database is a good way to store the data independently of the different ontologies that may be used to publish them. Indeed, it was easier for us to sketch a data model to store all required information (we knew we had to manipulate authors, books, etc.) but the exact definition of the exposed data model was a delicate issue. For instance, we internally define the notion of `Person` (e.g. *Victor Hugo*), which is later exposed as a *skos:Concept* and as a *foaf:Person* which share common properties but also have specificities. Besides a potential problem of data duplication, enforcing this duality in the data model would complicate the application code and logic since we nearly never have to make this distinction. Furthermore, both ontologies require different granularity of information. *foaf:Person* will need *foaf:name* and *foaf:surname* properties whereas *skos:Concept* will expose a concatenation of those properties in a *dc:title* field.

For those reasons, using a simple and strongly typed data model and storing data efficiently in a SQL database, allows us to program very easily with standard software components and libraries, and to publish data in whichever format is required (several ontologies, several output formats such as JSON, PDF, etc.). Of course, the internal data model changes regularly but CubicWeb provides helpers to do it very smoothly.

Avoiding Duplications. As stated above, another interesting aspect is that we avoid information duplication (which is still important, especially with millions of entities). Indeed, in the previous example, the same SQL records (author name and surname) are used for generating several RDF triples. The same thing is useful for works for example: works have a title that may be represented by a *dc:title* or a *skos:prefLabel*. Using an underlying SQL database avoid data duplication, as the two RDF triples are generated from only one SQL record.

Inner Model Can Be More Stable than Published Ontologies. Keeping the structured information in a SQL database, it is very easy to generate new RDF triples and push them into the graph. Moreover, changes in ontologies are easily handled by regenerating the RDF triples according to the new versions of the ontologies without using more complex tools of ontology evolution.

For example, let's consider the publication of the *BnF ontology* for the authors' roles. An author is related to a document (*Manifestation*) with a given role (*writer*, *scientific editor*, *trompetist*, ...). In the first versions of data.bnf.fr, these roles were published in the RDF using the *id.loc.gov* role referential, whereas in the HTML pages, the BnF's own roles referential was used, with a granularity that better fits its data. In recent versions of the Web site, the BnF referential has been published in RDF and is now used in the generated triples. The current architecture allows us to display the same information with a different granularity between the views.

All Internal Data Doesn't Have to Be Published. Every bit of data in the application is defined according to the internal data model, including *statistics*, *authentication data* from the LDAP directory, *title sort keys*. This information is needed to have a fully functional website but it doesn't make sense to publish them in RDF. Gathering all this data at the same place is definitely not necessary but it eases development and allows us to build simple query. E.g "Give me the 10 most visited documents, negotiated in RDF, in the last 5 days, sorted by number of visits, then by alphabetical order", and then apply the same `list` view that we can find on standard pages, in two lines of python code:

```
Any X, C ORDERBY C DESC, T LIMIT 10 WHERE H stats_about X, X title_sort_key T,
      H hit_type "rdf", H count C, H period P, P start > TODAY - 5
```

RQL Is not SPARQL. SPARQL is a great query language that has become the standard in the Semantic Web community. CubicWeb provides a simple SPARQL to RQL translator that transforms a standard CubicWeb application into a SPARQL endpoint. Unfortunately, only a subset of SPARQL is usable and only a subset of the internal data is queryable. This is partly because semantics of both languages differ a bit, but mostly because it requires an automatic mapping of the internal data model (defined in `Yams`, queryable in RQL) to the published data model, which is sometimes just not possible. For simple cases, CubicWeb uses a simple API to define equivalences or transformations between the internal `Yams` datamodel and the published RDF data:

```
# a Person should be translated into a foaf:Person
xy.add_equivalence('Person', 'foaf:Person')
# the surname property is transformed into foaf:familyName
xy.add_equivalence('Person surname', 'foaf:Person foaf:familyName')
# the birthplace is transformed into the placeOfBirth property
xy.add_equivalence('Person birthplace', 'RDAgroup2elements:placeOfBirth')
```

A very simple alternative would be generate all the rdf triples from the internal SQL database, push them in a triplestore and use CubicWeb `hooks` to keep the data up-to-date.

4.2 A General Overview of Involved Datasets

To interlink the data to other datasets, we use the fact that many semantic data are also open, and allow us to avoid to restart all the alignments from zero. The high quality of the alignments and the large number of authors make VIAF a crucial referential to be aligned with. It is also aligned on Wikipedia, and provides bridge between other reference databases. Using the VIAF's dump, we create **15937 exact matches** with the authors in our database. Moreover, this referential database is also a good way to benchmark the alignment tools used or developed in this project (see the *Data Alignment* section below). The matching are based on already existing alignments (e.g. VIAF, Geonames), and were derived using the URIs of the entities. Thus the disambiguation issue was considered as already solved in these dumps.

Thus, appart from the internal databases of the BnF, we use different external open databases and referentials:

`Dbpedia` **5488 exact matches** on Dbpedia, **3947 exact matches** on the french Wikipedia; `DnB` (German National Library) **26088 close matches**; `Geonames` **7038 close matches**, **22951 exact matches**; `Agrovoc` **685 exact matches**; `LCSH` **82937 close matches**; `Sudoc` **3318 exact matches**; `Thesaurus W` **66 close matches**, **979 exact matches**.

By aggregating different RDF dumps (nt, rdf/xml, CSV), and by performing simple string matching, we manage to create more **169290 (close/exact) matches** between the presented database and more than **8 different referential datasets**. Moreover, these databases allow a rapid and easy increase of the interlinking of our data, as they already present alignments to other database (e.g. Dbpedia is also aligned with Freebase, Project Gutenberg, New York Times ...)

4.3 Data Alignments

Many documents in the original catalog were not aligned on a *FRBR Work*: therefore we had to build such links. For example, the *FRBR Manifestation "Les Misérables, by Victor Hugo"*, should be aligned on the *FRBR Work "Les Misérables"* by the author *"Victor Hugo"*. The first approach is a *Naive alignment*. This alignment strategy is based on basic string matching, with few normalization pre-processings. It basically checks if two strings start similarly, while removing some common stopwords (e.g. *Le*, *Les*, ...):

- *"Misérables, Les, par Victor Hugo"* is aligned to the FRBR Work *"Les Misérables"*.
- *"Les Misérables, édition de 1890"* is aligned to the FRBR Work *" Misérables"*.

However, some cases are far more difficult and are not covered by the previously described business logic, e.g. *"La véritable histoire des Misérables, par Victor Hugo"*. For such cases, we develop a machine-learning based alignment that works on a bag-of-words representation of the *FRBR Manifestations* to be aligned, and on the *FRBR Works* of the author. Basically, we build a one-versus-all classification scheme based on Logistic Regression using the scikit-learn, for each of the *FRBR Work* of the author, in order to predict if a new *FRBR Manifestation* may be considered to be a representation of the Work or if it is not close enough compared to the other works. This approach allows to perform the following alignments:

- Multiple references to works, e.g. *"Les Misérables et Notre-Dame de Paris, Victor Hugo"*,
- Deletion/insertion of words, e.g. *"Notre-Dame, 1890"*, Les [1892] Misérables,
- Different words order/mispelling, e.g. *"Notre Dam de Paris"*, *"Paris, Notre-Dame de"*.

Finally, other kinds of data have to be aligned. For example, the Rameau subject *Nice* for the French city in the south of France should be aligned with the heading describing *Nice* and with some external referential such as Geonames. For such large-scale alignment (> 100.000 elements by corpus), we use the Nazca (live demo python library that provides a high-level API for data alignment, with SPARQL/RQL utilities.

4.4 URIs and URLs

In this project, we face different technological issues. One of them is the requirement of unique ids and stable URIs.

Ids Dequirements and Stable URIs. One of the input database of the project is the *BnF Archives et Manuscripts* database. However, as opposed to the main *BnF Catalogue General database*, we do not have unique ids/URIs to refer to those documents. We decided to use the *ARK* specifications [8] to automatically assign an id to the documents, based on the archives number.

We had to build URIs that are both stable (Semantic Web requirement) and human readable (so that an URI can be clearly related to the concept behind). The main difficulty here relies on the fact that the label used to describe the different concepts may change. Indeed, as a reference authority, the BnF chose a preferred way to label an author. However, this label may change, and using the label in the URI is thus conflictual. To solve this, we build URI of the form *http://data.bnf.fr/ark*

URL Redirection / Content Negotiation. We use the *ark* identifier system to identify each entity in a stable way, and to build the, *resource identifier URI* (following the cool URI conventions [11]) `http://data.bnf.fr/<ark-of-the-entity>`, that redirects to the *document resource URI* `http://data.bnf.fr/<notice-id>/<human-readable-title>`

The `notice-id` part of the URL is an internal, stable and unique identifier used at the BnF to index notices. This identifier is used as a seed to build the final `ark` identifier. The actual content delivered when asked for the *document resource URI* depends on the content negotiation step. Negotiable content types are `RDF` (`nt`, `n3` or `xml`), `PDF` and `HTML`. `Content-Location` header will be set accordingly.

For instance, the following HTTP request:

```
GET http://data.bnf.fr/ark:/12148/cb11928669t
     Accept:text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
     Accept-Charset:ISO-8859-1,utf-8;q=0.7,*;q=0.3
     Accept-Encoding:gzip,deflate,sdch
     Accept-Language:fr-FR,fr;q=0.8,en-US;q=0.6,en;q=0.4
```

will redirect to `http://data.bnf.fr/11928669/voltaire/`:

```
GET http://data.bnf.fr/11928669/voltaire/
     Accept:text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
     Accept-Charset:ISO-8859-1,utf-8;q=0.7,*;q=0.3
     Accept-Encoding:gzip,deflate,sdch
     Accept-Language:fr-FR,fr;q=0.8,en-US;q=0.6,en;q=0.4
```

that will in turn answer:

```
    Content-Encoding:gzip
    Content-Language:fr
    Content-Location:http://data.bnf.fr/11928669/voltaire/fr.html
    Content-Type:text/html;charset=UTF-8
```

5 Discussion and Applications

5.1 Building a Domain-Specific RDF Model with Standard Vocabularies

Libraries have a strong tradition of data exchange and interoperability, with the use of the *Marc* formats since the early seventies. But these library-specific formats are obviously limited to library communities, and hard to use for developers out of the "library world". By moving them to RDF we meant to facilitate new and unexpected uses of our data from different communities. That is why we chose common, simple vocabularies that are widely used and tested on the Web: `skos` [12], to describe concepts; `foaf` [13] for persons and organizations; existing library vocabularies such as `Dublin Core` [7] for bibliographic information. We also used the ontology `bnf-onto` only for classes and properties that were not expressed anywhere else:

http://data.bnf.fr/ontology/bnf-onto/. All these library-specific terms were declared as sub-properties or sub-classes of existing vocabularies, for those who want broader information. For example http://data.bnf.fr/ontology/bnf-onto/ouvrageJeunesse to sort adapted editions of a work for the younger public, and which is a subclass of DCMIterms: text.

Though we tried to have a simple model for end users, we also experiment on the data models that are currently being discussed in the library community. We use the *Functional Requirements for Bibliographic Records* (FRBR model) to express precise and relevant links between our data. This model is specified at http://data.bnf.fr/semanticweb (French) and http://data.bnf.fr/semanticweb-en. We also used and published BnF specific vocabularies (http://data.bnf.fr/vocabulary) that are matched to the Library of Congress: Country codes list, Relator codes list, Types of subject headings.

5.2 Retrieving Data: Simple or Extensive Re-use

Together with the serializations that we provide, namely RDF/XML, NT, N3, several users asked for a simplified view with the main concepts, the links to the digitized documents, without the whole descriptions of every document. Therefore we offer a JSON view with a simplified model and less resources. There already are developers of small applications, who build timelines for research purposes- or for smartphone applications.

We wanted our data not only to be used on the Web, but also to be visible, in order to reach a new public that do not know about the BnF collections. As *author*, *work* and *subject* pages are open on the Web and can be reached by search engines, we provide HTML embedded data from Schema.org. These elements are used by search engines to identify, disambiguate terms, and, above all, to put forward digital documents. As a consequence, Gallica pages can be easier to find when they are in data.bnf.fr. We also integrated Opengraph Protocol (OG) metadata, so that the pages can be represented in social networks. Adopting these vocabularies answers to a different logic than displaying our data in RDF: as schema.org vocabularies have been created by and for search engines, they are simple and have a high level granularity. The library follows the evolutions of Schema.org for libraries (http://www.w3.org/community/schemabibex/).

Users can download data in RDF-XML, NT or N3, either for each page, through content negotiation or by clicking on the RDF logo, or get a bulk download of all data. As the volume of data is progressively increasing, the site is going through performance difficulties: the RDF pages as well as the RDF dump are hard to generate. So we decided to lighten the RDF of the pages, to attribute actionable URI to manifestations and to split our dump, according to the different uses of our data: lighter dumps for authors, works, subjects, a dump with complete detailed manifestations, and one for external links. To go further, users want to pick and choose the data they need without necessarily downloading everything: for example, taking only the main information about the author (his different names, his dates for instance), and leaving aside the documents to which he contributed. Therefore the next step could be to offer a SPARQL endpoint, to enable a dynamic interrogation and retrieval of our data to the external user, but also for the BnF operators to have a better knowledge of its own data. This idea of getting to know our collections better through Semantic Web technologies is very important, because it makes the improvements important on the business side, not only in terms of services for the end-users, but also in terms of curation of our collections, in a long-term perspective.

5.3 Enhancing Services in the Long Term

Providing structured data in Open data enables to create new links and new interfaces. We had feedbacks from end-users, mainly directly on the interface, and through statistics on use (search by title of a work for instance). Since all the raw data is available with an Open License, we had comments from developers, or instance on the properties and vocabularies used, or to ask for a SPARQLendpoint. We take these remarks into account as the site is being developed. As the volume of the base is increasing, we may provide new services such as a SPARQL endpoint. We can also have an idea of the kind of content that is being used and how. Some of end-users are re-distributing the dataset and referencing it for others to re-use, starting with `data.gouv.fr`, the official Open data portal of the French State, but also other sites such as CKAN, OKF and Open data directory. Other users are data specialists from the cultural sector, who use a part of the data for specific purposes in their local applications, such as the `Institut français`. This broad range of uses of the "raw data" shows us that library information can be useful for broader communities. Some users now can avoid duplication of data when indexing resources. For example, displaying authority data, such the thesaurus RAMEAU in SKOS, with over 160 000 subjects, was very much expected by users. The project *MACS* (Multilingual access to subjects) [14] is a good use case in that matter. It matches subject authority from the BnF (RAMEAU), the Deutsche Nationalbibliothek, the Library of Congress and the National Library of Switzerland so that users can put the keywords they chose in their own language.

The BnF also tries to enhance new uses of its data. RDF allows the library to create new services, by following the links in RDF graph. For instance, you can provide all the different editions of a work in a digitized version. These links can be used in apparently simple functionalities, such as: finding other editions of a book, digital versions of it, other works by a writer, and so on.

Finally, combining Semantic Web tools with matching techniques, we answer to a great demand inside the library to improve the catalogues at the source without increasing the cataloguers' work. It is of course useful inside the application. But it can also be used in the original library catalogues. Thus, if data.bnf.fr may lead to new services for the external users, it is also a way to improve and correct automatically or semi automatically our own data, in the long term. That is why we try to build routines and mechanisms that can be used inside the original catalogues. Step by step, inferences on our data are integrated to the original sources. A specific interface has been developed on the business side, so that librarians can validate the automatically generated matchings. We can automatically generate *Work* pages inside our authority files or create new links between bibliographic records and the work authority file, following the FRBR principles. The aim is to reduce cataloguing tasks as much as possible, and to create new links, in a way that can be immediately useful for the end-user.

6 Conclusion

Semantic Web tools made our data visible and exchangeable on the Web. As we are manipulating data and not bibliographic records, we could imagine new ways of organising the information, such as pages for a date (e.g. http://data.bnf.fr/what-happened/date-1515) or for a role (example: all the authors who have been making coins, such as *Louis XIV*, http://data.bnf.fr/vocabulary/roles/r370). The software may also display graphic visualization of the data, such as maps, diagrams or time-lines, and bring new opportunities, about the use of geographical data for instance.

Today (March 2013), http://data.bnf.fr displays millions of RDF triples, corresponding to 20 % of the BnF main catalogue. The next step is to increase gradually the volume to include the whole main catalogue, with its 1.6 million authors, in the long term. This will imply performance issues, but also a real opportunity to bring valuable and massive data on the Web. Correlated with matching techniques and data mining, Semantic Web is a condition and an opportunity to create new links and new services in interfaces that have to remain easy to use and quick to understand.

References

1. Martin, F.: Gallica, bibliothèque et plate-forme numériques, 12e Journées des Pôles associés et de la Coopération (2009)
2. Bermès, E.: Des identifiants pérennes pour les ressources numériques. l'expérience de la bnf (2006)
3. Crawford, W.: MARC for library use: understanding integrated USMARC. Professional librarian series. G.K. Hall (1989)
4. Taylor, A., Miller, D.: Introduction to Cataloging And Classification. Library and Information Science Text Series. Libraries Unltd Incorporated (2006)
5. Stührenberg, M.: The tei and current standards for structuring linguistic data an overview. Journal of the Text Encoding Initiative (3) 1 – 14
6. Pitti, D.V.: Encoded archival description: An introduction and overview. D-Lib Magazine **5**(11) (1999)
7. Dekkers, M., Weibel, S.: Dublin core metadata initiative progress report and workplan for 2002. D-Lib Magazine **8**(2) (2002)
8. Kunze, J.A.: Towards electronic persistence using ark identifiers, ark motivation and overview (2003)
9. Baker, T., Bermès, E., Coyle, K., Dunsire, G., Isaac, A., Murray, P., Panzer, M., Schneider, J., Singer, R., Summers, E., Waites, W., Young, J., Zeng, M.: Library linked data incubator group final report: W3C incubator group report 25 october 2011. Technical report (2011)
10. on the Functional Requirements for Bibliographic Records, I.S.G.: Functional requirements for bibliographic records : final report. (2009)
11. Sauermann, L., Cyganiak, R., Völkel, M.: Cool uris for the semantic web. Technical Memo TM-07-01, Deutsches Forschungszentrum f ür Künstliche Intelligenz GmbH (2007)
12. W3C: SKOS Simple Knowledge Organization System Reference. (2009)
13. Brickley, D., Miller, L.: FOAF Vocabulary Specification 0.97. Namespace document (2010)
14. Landry, P.: Multilingual subject access: the linking approach of MACS. Cataloging & Classification Quarterly **37**(3/4) (2004) 177–191