Undefined 0 (0) 1
IOS Press

Multilingual Linked Open Data Patterns

Jose Emilio Labra Gayo [a,*], Dimitris Kontokostas [b] and Sören Auer [b]
[a] *Department of Computer Science. University of Oviedo, CP. 33007, Oviedo, Spain*
Email: labra@uniovi.es
[b] *Universität Leipzig, Institut für Informatik, AKSW, Postfach 100920, 04009 Leipzig, Germany*
E-mail: {lastname}@informatik.uni-leipzig.de

Abstract. The increasing publication of linked data makes the vision of the semantic web a probable reality. Although it may seem that the web of data is inherently multilingual, data usually contains labels, comments, descriptions, etc. that depend on the natural language used. When linked data appears in a multilingual setting, it is a challenge to publish and consume it. This paper presents a survey of patterns and best practices to publish Multilingual Linked Data and identifies some issues that should be taken into account. As a use case, the paper describes the patterns employed in the DBpedia Internationalization project.

Keywords: Linked Open Data, Multilingual Web, Language tags, Multilingual Data

1. Introduction

The linked data paradigm is based on a small set of best practices which were outlined in [10]:

- Use URIs as names for things;
- Use HTTP URIs so that people can look up those names (aka. dereferencing);
- Return useful information when upon lookup of those URIs (esp. RDF);
- *include links* by using URIs, that dereference to other documents.

Since the publication of those guidelines, an increasing number of projects have been devoted to the publication of linked (open) data, making the vision of a semantic data web more plausible [13].

The principles of linked data have been described in several publications and books [12,30]. There is even a book devoted to linked data patterns [18].

Nevertheless, most of the projects that publish linked data employ English as the primary language. In fact, most popular vocabularies contain descriptions and labels only in English.

A study by Ell et al. [24] has shown that language tags are a rarely used feature in popular datasets. Only 4.7% of the non-information resources employ one language tag, and only 0,7% employ several language tags. It was discovered that the most popular language tag was English (44.72%), while the rest of the languages had a very low appearance: German (5.22%), French (5.11%), Italian (3.96%), etc. As can be seen, this distribution does not reflect the number of people that speak those languages, or even the number of web pages written in them.

The lack of multilingualism in linked data can be attributed to its recent birth. However, another reason may be that there is a lack of design patterns and guidelines to help the community that wants to publish multilingual linked data.

In this paper, we collected, justified and explained a comprehensive set of those patterns. It must be noted, that we do not propose the patterns as *best practices*. In most of the cases, the proposed patterns offer a possible design decision with pros and cons that must be taken into account. The aim of this paper is to clarify the benefits of those decisions.

*Corresponding author. E-mail: labra@uniovi.es.

In this way, the main contribution of this paper is to collect a catalog of design patterns related to multilingual linked open data. For each pattern we provide a short description, the context where it can be applied, a motivating example, a discussion of the pros/cons, and we also relate that pattern to other patterns.

We assume that the reader is familiar with basic RDF and SPARQL concepts. We employ the Turtle notation [4] along the paper. In order to simplify the examples, we omit prefix declarations[1].

The paper is structured as follows. Section 2 contains a short overview of the main concepts related to internationalization of semantic web technologies. In section 3 we define the concept of multilingual linked open data and give a running example that we will employ along the rest of the paper. Section 4 contains the proposed catalog of multilingual linked open data patterns. Section 5 describes DBpedia International as a use case of a multilingual linked open data solution, we describe and justify the solutions taken. Finally, we describe related work in section 6 and outline conclusions and further work in section 7.

2. Internationalization and Web Architecture

One of the benefits of the World Wide Web is the development of a global information system which can be used by any person independently of its language. This principle has been described by the World Wide Web Consortium (W3C) as the *Web for All* principle: *The social value of the Web is that it enables human communication, commerce, and opportunities to share knowledge. One of W3C's primary goals is to make these benefits available to all people, independent of hardware, software, network infrastructure, native language, culture, geographical location, as well as physical or mental ability.*[2]

In fact, the first WWW Conference in 1994, included a paper about multilingual information exchange [45] and in 1995, an activity was started by the W3C towards more internationalization of the Web. Currently, the *W3C Internationalization Activity*[3] maintains a large number of resources on the different aspects of WWW internationalization.

2.1. Characters and Unicode

One of the basic aspects of internationalization is character representation. Initial standards such as ASCII were based on the Latin alphabet. Many technologies were and still are based on ASCII character encoding. However, with the increasing adoption of computers, the need to increase the character repertoire became obvious.

Unicode started in 1987 as a project of the Unicode Consortium to develop a universal character set. The *International Standards Organization* (ISO) also chartered a working group with the same goal called ISO/IEC 10646. In 1991 both institutions worked together to synchronize both specifications.

The goal of the Unicode project is to cover all characters from all writing systems of the world, modern *and* ancient. It also includes technical symbols, punctuations, and many other characters used in written text.

In Unicode, each character is assigned a code point. It is convenient to distinguish between the code point and the glyph of a character. A code point is the number assigned to a character while a glyph is mainly the particular image representing a character or set of characters.

Some code points may have the same glyph. For example, the character `o` in the Latin alphabet corresponds to the code point `0x007F` (LATIN SMALL LETTER O) but the same glyph `o` corresponds to `0x0585` (ARMENIAN SMALL LETTER OH).

In other cases, the same glyph can be obtained by combining several characters. For example, the letter `ñ` can be directly represented as `0x00F1` (LATIN SMALL LETTER N WITH TILDE) or by two code points: `0x004E 0x0303` (LATIN SMALL LETTER N, COMBINING TILDE).

Unicode proposed several solutions called normalization forms to avoid these ambiguities and to check whether two sequence of Unicode characters are equivalent.[4]

[1] We use popular aliases which can be expanded using the `http://prefix.cc` service

[2] `http://www.w3.org/Consortium/mission`

[3] `http://www.w3.org/International/`

[4] `http://www.unicode.org/charts/normalization/`

In the case of the Web architecture, the solution adopted was *Normalization Form Canonical Composition* (NFC) and is described in [3]. The use of normalization is important for linked data as it is necessary to have a non-ambiguous way to check if two identifiers are the same.

2.2. URIs and IRIs

One of the cornerstones of Web architecture is the use of Uniform Resource Identifiers (URIs) [8] to identify any kind of resource. URIs were formally described by the IETF RFC 3986 [11] and their design offered a trade-off solution between readability and usability.

The readability goal can be achieved when URIs are easily remembered and interpreted by people, employing meaningful or familiar components. To that end, people from non-Latin alphabets should be allowed to use their own alphabets (i.e. Unicode characters) in their URIs. However, due to the usability design trade-off, the URI specification restricted the character repertoire to US-ASCII characters for easier transmission and storage in legacy systems

The use of non-ASCII characters in URIs can be achieved by percent encoding the octets of the character, decreasing the ability to read and remember those URIs.

For instance, the Spanish letter "ó" is percent encoded to "%F3". In this way, the name of a city like León is encoded as Le%F3n. This encoding can be still readable when there are few extended characters, but it is clearly unreadable when the word is completely formed by non-Latin characters.

IRIs [22] were designed to identify resources using Unicode characters. An example of an IRI is: http://españa.es/León which could identify the Spanish city León. Note that it is possible to use Unicode characters in both the domain name españa.es and the path León. Although IRI supported increases incrementally, there is still a number of protocols and systems that only accept ASCII characters or accept Unicode characters only partially.

The conversion from IRIs with Unicode characters to ASCII-only URIs is performed in two parts: the path and the domain name. The path is percent encoded using UTF-8 characters as we have shown in the previous example. In the case of domain names, the conversion employs an algorithm called *punycode*, which efficiently converts characters between Unicode and ASCII. As an example, the domain name españa is punycoded to xn−−espa−jqa. Browsers supporting punycode automatically and convert the IRI to its punycode representation. In order to prevent homograph attacks (i.e. IRIs whose glyphs look the same but contain different characters) browsers can display the original IRI or the punycode representation depending on the user language preferences.

2.3. Languages and the Web

From the beginning, the architecture of the Web has taken the existence of different languages and the need for language-aware protocols and specifications into account. As an example, HTML contains the lang attribute to specify the base language of a portion of HTML code. XML also offers the xml:lang attribute with the same purpose. In the same vein, RDF also contains literals which can be associated with language tags as we will show in the next section.

At the protocol level, HTTP provides the Accept−language header field, where a user agent can restrict the set of natural languages that are preferred as a response to a request. Header fields can be used for HTTP content negotiation, where it is possible to obtain different representations of the same resource depending on those fields.

In the case of linked data, it is a well-known practice to dereference a URI and return different representation formats (HTML, RDF/XML, JSON, etc.) depending on the Accept header.

Language declarations must employ IETF language tags defined in BCP47 [41], like en (English) or es−419 (Latin American Spanish). It is important to clearly distinguish between countries and languages. For example, to identify Armenian, one should use hy (Hayastan) instead of am which identifies the country Armenia but the language Amharic spoken in Ethiopia.

3. Multilingual Linked Data

A rationale of the Web of Data is to develop technologies that enable machines to consume data. Although one may consider that data is intrinsically multilingual, we can see that data usu-

ally contains references to textual information in some natural language.

As a running example, imagine that we want to declare the following information:

> *"Juan is a professor at the University of León. He was born in Armenia and is 43 years old"*

In that paragraph, there is data that does not depend on multilingualism, like the age, so it can be considered *intrinsically* multilingual, and can be represented as:

```
:juan :age "43"^^<xsd:integer> .
```

However, most of the data also contains references to human-readable textual information. For example, if we want to declare that the position of :juan is Professor, we can use:

```
:juan :position "Professor" .
```

Now, that data is language dependent and if our application is meant to work with other languages, we may want to attach localized literals.

For example, if we are planning to translate the linked data application to Spanish we may be interested to declare that *Juan* is *Professor* (in English) but *Catedrático* (in Spanish).

This simple example can be solved by employing language-tagged literals that correspond to the *multilingual labels* pattern (cf. section 4.3.2) that we will present later.

```
:juan :position "Professor"@en .
:juan :position "Catedrático"@es .
```

Definition. *Multilingual data* is defined as data that appears in a multilingual setting and contains references to human readable textual information in several languages.

In the context of this paper, we are interested in multilingual linked data, which is multilingual data that follows the linked data principles.

Some examples of multilingual linked data are international vocabularies like *Agrovoc* or multilingual datasets like *DBpedia* (cf. section 5).

In general, almost all linked data can be considered multilingual linked data when its purpose is to serve multilingual communities. For example, DBpedia contains linked data which can be used by a large number of international users.

When publishing multilingual linked data it is necessary to take into account that future applications of that data may need to be localized to different environments. Multilingual linked data should be published in a way that enables that process. In the following section we will review the main patterns related to multilingualism and linked data.

4. Multilingual Linked Open Data Patterns

In this section, we present a catalog of patterns for implementing multilingual linked data solutions. The patterns are classified according to the common tasks that have to be performed to publish linked data:

- *Naming.* This task refers to the process of URI design and dataset description from a multilingual point of view.
- *Dereferencing.* This task describes patterns to handle dereferencing in a multilingual environment. For example, should we return the same or different representation depending on the language?
- *Labeling.* When publishing linked data, there are a number of reasons to label different resources. In a multilingual setting, how can we provide labels for different languages?
- *Longer descriptions.* We consider that resource labels are different from longer textual information. Not all textual information attached to resources are labels and in fact, longer descriptions, like comments, or even book chapters, can be encoded in literals and contain textual information in different languages.
- *Linking.* In a multilingual setting, it is possible to have different resources representing the same thing in different languages. How can we link those resources?
- *Reuse.* Linked data is all about reuse of data. When multilingual data is linked to vocabularies, is it better to have multilingual vocabularies or to localize existing ones?

Figure 1 gives an overview of the proposed catalog of patterns which are described in the following sections.

Table 1
Overview of Multilingual LOD patterns

Classification	Name	Description	Section
Naming	Descriptive URIs	Use descriptive URIs with ASCII characters, % encoding extended characters	4.1.1
	Opaque URIs	Use non human-readable URIs	4.1.2
	Full IRIs	Use IRIs with unicode characters	4.1.3
	Internationalized Local names	Use Unicode characters only for local names	4.1.4
	Include language in URIs	Include language information in the URI	4.1.5
Dereference	Return language independent data	Return the same triples independently of the language	4.2.2
	Language content negotiation	Return different triples depending on user agent preferences	4.2.1
Labeling	Label everything	Define labels for all the resources	4.3.1
	Multilingual labels	Add language tags to labels	4.3.2
	Labels without language tag	Add labels without language tags in a default language	4.3.3
Longer descriptions	Divide long descriptions	Replace long descriptions by more resources with labels	4.4.1
	Lexical information	Add lexical information to long descriptions	4.4.2
	Structured literals	Use HTML/XML literals for longer descriptions	4.4.3
Linking	Identity links	Use owl:sameAs and similar predicates	4.5.1
	Soft links	Use predicates with soft semantics	4.5.2
	Linguistic metadata	Add linguistic metadata about the dataset terms	4.5.3
Reuse	Monolingual vocabularies	Attach labels to vocabularies in a single language	4.6.1
	Multilingual vocabularies	Prefer multilingual vocabularies	4.6.2
	Localize existing vocabularies	Translate labels of existing vocabularies	4.6.3
	Create new localized vocabularies	Create custom vocabularies and link to existing ones	4.6.4

4.1. Naming

It is a well known best practice that URIs should not change [9] and should not depend on implementation techniques, file name extensions, etc. In order to obtain stable URIs, the first step in a linked data development lifecycle is to design good URI schemes [15].

4.1.1. Descriptive URIs

Description. Descriptive URIs use ASCII characters that are combined to represent terms or abbreviations of terms in some natural language. It is usually done with terms in English or in other Latin-based languages, like French, Spanish, etc. where only a small fraction of their alphabets is outside ASCII characters.

Context. Descriptive URIs are appropriate when most of the terms are in English or in Latin-based languages. It can be also applied when interoperability with existing systems is vital.

Example 1. An example of a URI that represents Armenia could be:

```
http://example.org/Armenia
```

Discussion. The characters that appear in a URI usually represent natural language terms to improve human-readability. Using simple URIs in ASCII has the advantage that ASCII characters are very well supported by almost any computer system. This pattern offers a good balance between readability and usability of resource identifiers. However, for most languages other than English, the natural script usually contains characters outside of ASCII. In the case of languages with completely non-Latin scripts (Armenian, Arabic,

Greek, etc.) ASCII only URIs are very restrictive and percent-encoding local names renders them unreadable.

See also. Descriptive URIs are also called meaningful URI local names in [40], although they refer only to the local name part of the URI. This pattern is opposed to the *opaque URIs* pattern (4.1.2). When the restriction of only ASCII characters is removed, this pattern becomes the same as *Full IRIs* (4.1.3) or *Internationalized local names* (4.1.4). It is related to the *URL slug* [5] pattern in [18] where URIs are generated from text of keywords.

4.1.2. Opaque URIs

Description. Opaque URIs are resource identifiers which are not intended to represent terms in a natural language.

Context. Opaque URIs are a good solution when most of the resources are obtained automatically from other systems like relational databases, tables, etc. This pattern can also be applied when there is a need to have multilingual concepts and it is preferred not to have any language bias. Using opaque URIs may help to separate the concept from its different labels.

Example 2. An opaque URI can be:

```
http://example.org#I23AX45
```

Discussion. Opaque URIs can help to emphasize the independence of a resource from its natural language representation. This pattern emphasizes that URIs are not meant for end users but for internal applications. Depending on the context, URIs with human-readable local names can help users and application developers to manage and debug linked data applications. Resources with opaque URIs usually depend on labels to hide the URI from the end user.

See also. Opaque URIs are described in [40]. There are some well-known vocabularies that employ opaque URIs, like EuroWordnet [46] or Agrovoc [14]. This pattern is associated to the *Label everything* (4.3.1) pattern so that the application can show some information to the end user.

[5] http://patterns.dataincubator.org/book/url-slug.html

This pattern is also related to the *Natural Keys*[6] pattern in [18], in the sense that Opaque URIs are often the result of a mapping process between some external identifiers which are algorithmically converted to linked data URIs.

4.1.3. Full IRIs

Description. This patterns consists of using unrestricted IRIs which can contain Unicode characters outside the ASCII repertoire.

Context. In a multilingual setting, it is necessary to take into account that human-readability is not a generic aspect, but depends heavily on people's culture and background. URIs employing only ASCII characters are difficult to handle by people used to non-Latin alphabets. This pattern can solve that situation by allowing the application to use Unicode characters in resource identifiers.

Example 3. A full IRI using the Armenian language can be:

```
http://օրինակ.օրգ#Հայաստան
```

Discussion. IRIs with Unicode characters are more natural for people whose primary language is not Latin based. Since machines should be able to identify resources in either encoding and the technologies have already been developed, a further step is to make resource identifiers human friendly. Although it is said that the end user should not be exposed to URIs and that they should act as internal identifiers, in practice, they are handled by application developers and sometimes even by end users.

Human friendly IRIs can facilitate the adoption of linked data technologies by more people in the long term. However, as explained in section 2.2, the use of IRIs may be exposed to visual spoofing attacks given that glyphs with the same appearance may refer to different characters.

Another important issue is the lack of support of IRIs by current software libraries. Although the support is improving, nowadays it is still a challenge [7] and most of the tools only offer partial support.

[6] http://patterns.dataincubator.org/book/natural-keys.html

See also. Unicode has published some security considerations for IRIs which should be taken into account [19]. A soft version of this pattern is *Internationalized local names* (4.1.4). According to [34, section 6.1], IRI dereferencing should be handled carefully as the HTTP Protocol (RFC 2616) [26] can transfer only URIs.

4.1.4. Internationalized local names

Description. Internationalized local names are IRIs where the domain part is restricted to ASCII characters while the local name can use Unicode characters.

Context. This pattern can offer a trade-off between security and readability. On one hand, it limits ASCII characters for the domain part, which may be subject to homograph attacks. On the other hand, the use of local names with Unicode characters improves readability.

Example 4. Armenia can be identified with this hybrid approach by:

```
http://example.org#Հայաստան
```

Discussion. This pattern avoids the problems associated with domain name spoofing while it offers more human-friendly resource identifiers. While this partially solves the problem, the possibility of spoofing using visual equivalent IRIs for different purposes remains. However, as the domain name is preserved as the authoritative source, it is much more difficult to accomplish such attacks.

See also. This pattern is opposed to the *Opaque URIs* pattern (4.1.2). It extends the *Descriptive URIs* pattern (4.1.1) but is more restrictive than the *Full IRIs* pattern (4.1.3). It is employed in DBpedia International (cf. Section 5).

4.1.5. Include language in URIs

Description. This pattern proposes to insert a language identifier in the URI. Thus, datasets of different languages can be easily recognized by the URI.

Context. It can be applied when datasets are clearly separated by language. In this way, the different datasets may be generated, and even maintained by different servers which publish their corresponding language dependent datasets separately.

Example 5. The Armenian version of the country Armenia could be:

```
http://hy.example.org#Հայաստան
```

where hy represents the Armenian language (Hayastan). All the triples in Armenian could be hosted in hy.example.org while the triples in other languages, for example Spanish, could reside in es.example.org.

Discussion. In a multilingual setting, being able to easily recognize the language of a resource may facilitate the development. A more practical benefit is to separate different datasets, which can be obtained from different sources, by language.

However, notice that the employment of a language tag in the URI may contradict the *Cool URIs* philosophy in the sense that we are encoding extra information in the URI that may be subject to changes in the future.

Adding languages to the URI can become unwieldy if we consider sub-languages, dialects and regions. There are more than 7000 languages already registered[7] which can be very specialized. For example, hy−Latn−IT−arevela represents eastern Armenian written in Latin script as used in Italy. Including such a detailed information in the URI may not be reasonable.

Another design decision is where to put the language tag.

Example 6. It is possible to have alternative URI schemes depending on where we include the language tag in the URI.

```
http://example.org/hy#Հայաստան
http://example.org/Հայաստան/hy
```

The last pattern is less convenient as it mixes the Unicode characters of the local name with the ASCII characters of the language tag.

See also. This pattern can be combined with the *language content negotiation* pattern (4.2.1). It is possible to have a language agnostic URI for a concept without language tag and to use HTTP language content negotiation to redirect to the preferred dataset. This pattern is also related to the

[7] http://www.iana.org/assignments/language-subtag-registry

Patterned URIs[8] pattern in [18] in the sense that URIs are defined to follow a naming pattern (including the language). It is also related to *Hierarchical URIs* [9] of the same book.

4.2. Dereference

The dereference of a URI is the retrieval of the representation of the resource identified by that URI. Although this process seems orthogonal to multilingualism, the HTTP protocol includes the possibility of content negotiation in which the server can return different representation depending on the preferences of the user agent. This process can also involve language preferences. In general, there are two possibilities which are identified in two opposite patterns: return different representations depending on language preferences, or return always the same representations.

4.2.1. Language-based content negotiation

Description. In this pattern, the server attends the language preferences of the user agent, presented in the Accept−language header and returns different data for each language preference.

Context. This pattern can be used to reduce network bandwidth and client processing. Notice that a user agent retrieves a subset of all the triples in languages that he has included in the header.

Example 7. Imagine that the server contains the following triples

```
:juan :position "Professor"@en .
:juan :position "Catedrático"@es .
:juan :workPlace "León Universiy"@en .
:juan :workPlace "Universidad de León"@es .
```

If we try to obtain those triples from a server, which does language-content negotiation, and the header contains Accept−language:es then the server returns:

```
:juan :position "Catedrático"@es .
:juan :workPlace "Universidad de León"@es .
```

while if the header is Accept−language:en, it returns:

```
:juan :position "Professor"@en .
:juan :workPlace "León Universiy"@en .
```

Notice that the semantics of the HTTP content negotiation mechanism should return the whole dataset, if there is no language preference.

Discussion. This feature reduces network traffic and the load of client applications as the server would only send triples of a given language.

Content negotiation is part of the Web architecture. This solution can improve the performance of clients limiting the number of triples in languages that they are not interested.

Implementing language content negotiation complicates the development. Another problem is that if we return different representations in different languages, we should ensure that the content represented in one language is equivalent to the content represented in other language. This can raise problems on semantic equivalence between natural language text.

See also. This pattern is related to the content negotiation exposed as a best practice recipe for publishing RDF vocabularies in [1] where language content negotiation is applied to offer different representations for HTML or RDF content. Although this pattern is meant for human oriented representations, like HTML, for RDF there is no such specification. There are non-standard implementations by triple-stores (like the *sql:BEST_LANGMATCH* function in Virtuoso) that can provide the above mentioned functionality.

4.2.2. No language negotiation

Description. In contrast to the *Language-based content negotiation* pattern (4.2.1), this pattern returns always the same triples without taking into account the Accept−language header.

Context. RDF is meant for machines which are natural language agnostic, so ignoring the language preferences seems a reasonable option. This pattern can be employed when the amount of multilingual data is manageable and there are no big constraints on datasets consumers.

Discussion. Implementations that always return the same data could be considered more consistent. Ignoring the Accept−language header and returning all the information available for a given

[8] http://patterns.dataincubator.org/book/patterned-uris.html

[9] http://patterns.dataincubator.org/book/hierarchical-uris.html

resource seems a valid solution for software agents that will select the triples that are of interest to their end-users.

With this pattern the data representation of a resource offered is independent of the language preferences, so there is no need to care about semantic equivalence of textual information.

However, localized client applications may receive unnecessary triples that could create a computation and network overhead. This overhead could have an impact in low computation and bandwidth devices such as smart-phones and sensors.

See also. This is the opposite pattern to 4.2.1.

4.3. Labeling

Applications based on linked open data will need to expose data to the end user. It is a common practice to associate labels to resources so the application can show those labels to the end user.

The most accepted property for displaying labels is rdfs:label, although there are other possibilities like skos:prefLabel, dc:title, etc.

In this section we discuss the different patterns related to labeling and multilingualism. We separate *labeling* from *longer descriptions* in the sense that labels are short textual information attached to a resource. Labels could be considered as units of textual information.

4.3.1. Label everything

Description. Linked data datasets should provide labels for all resources: individuals, concepts and properties, not just the main entities.

Context. Linked data applications that contain data which is supposed to be exposed to an end-user in some natural language.

Although URIs may be human-readable, they are not expected to be seen by the end user. In order to improve user experience it is necessary to expose data and entities in human-readable ways. Labels facilitate:

1. displaying data to end-users, instead of URIs
2. searching over the Web of Data
3. indexing purposes, training, use of annotation tools, etc.

Discussion. In general, associating labels is a good idea. It is always better to offer a textual information to the end user than a URI. However, in some applications it can be difficult to find the right label for a resource, specially when resources are automatically generated.

When labeling resources, one must take into account that the purpose of those labels is mainly for humans. Using camel-case or similar notations should be avoided and the use of uppercase, space delimiters etc. should be consistent [40].

See also. [24] enumerates some uses for labels and contains a thorough study on the use of labels in popular datasets. This pattern also appears with the same name in [18] where dataset creators are urged to "Ensure that every resource in a dataset has an rdfs:label property".

4.3.2. Multilingual Labels

Description. In a multilingual setting, it is necessary to attach language tags to textual information, in order to identify the appropriate label for localized applications.

Context. This pattern can be applied when labels have information in some natural language.

Example 8. We can declare that Juan's position is *Professor* in English and *Catedrático* in Spanish.

```
:juan :position "Professor"@en .
:juan :position "Catedrático"@es .
```

Discussion. Multilingual labels are part of the RDF standard and well supported by semantic web tools. For example, the following SPARQL query asks for people whose position is Professor and would return :juan.

```
SELECT * WHERE {
  ?x ex:position "Professor"@en .
}
```

Dealing with big multilingual linked data repositories where all literals have a language tag makes SPARQL queries more verbose. The following SPARQL query for example 8 would return no results.

```
SELECT * WHERE {
  ?x ex:position "Professor" .
}
```

There are some patterns and built-in SPARQL functions that deal with language tagged literals like:

- lang() returns the language of a literal
- langMatches() checks if the language of a literal matches a given language or
- str() returns the literal without the language tag

Using the str() function, the previous query can be rewritten as:

```
SELECT * WHERE {
  ?x :position ?p .
  FILTER ( str(?p)="Professor" )
}
```

See also. This pattern is the same as the *Multilingual literal* pattern [10] in [18]. This pattern is also related to the *Repeated property* pattern where a resource can have several values for the same property. In this case, the values are literals in different languages.

4.3.3. Labels without language tag

Description. Apart of language-tagged labels, one can also associate plain labels without a language tag.

Context. This pattern emerges as a practical advice to facilitate SPARQL queries over multilingual linked data.

Example 9. Using a language without language tag, example 8 can be expressed as:

```
:juan :position "Professor"@en .
:juan :position "Catedrático"@es .
:juan :position "Professor" .
```

Discussion. Using this pattern SPARQL queries do not need to be aware of the multilingualism of the :position property and can obtain a result.

Although this practice facilitates SPARQL queries, it is controversial and can be also considered to be an anti-pattern. For instance, in which language should the literal without language tag be written? That would depend on the language spoken by the majority of the linked data users. In most cases it could be English but in other contexts it could be quite a different language.

[10] http://patterns.dataincubator.org/book/multi-lingual-literal.html

See also. The use of labels without language tags has been proposed by Richard Cyganiak in [17]. This pattern can be associated with the *language in URI* pattern (4.1.5), so that there is a hint on what is the default language of a dataset. It is possible to make assertions about the language of a whole dataset using the lexvo:language property which we will present in the *Add linguistic metadata* pattern (4.5.3).

4.4. Longer descriptions

Although labels can be really helpful in the case of human-enabled applications, there are cases where they do not suffice. Labels are meant to be very short descriptions of a resource. Depending on the context, short descriptions may cause ambiguity and thus, a better description may be required.

There are ways to provide longer descriptions for a resource. For example, there are some common properties (like `dcterms:description`, `rdfs:comment`, etc.). In other occasions, it may be better to alter the resource model or to add metadata to those descriptions. In this section we cover the main patterns related to longer descriptions.

4.4.1. Divide long descriptions

Description. By decomposing long descriptions into new resources that can be represented with shorter labels or lexical entities, one can facilitate their future translation to other languages.

Context. Resources with very long descriptions are usually a symptom that the knowledge base structure (i.e. vocabulary or ontology) is not sufficiently developed. In general, it is good to foresee to associate short labels with resources. This way, a model where resources have long descriptions can be further decomposed into small pieces with shorter descriptions or labels. Multilingual applications with more fine grained textual information can be better localized and adapted to other languages.

Example 10. In order to declare that Juan is a professor from the University of León, one can assert: The following triple represents the job title of Juan:

```
:juan :jobtitle
  "Professor at the University of León"@en .
```

The description can be decomposed in two components: the title (*Professor*) and the University (*University of León*).

```
:juan :position :professor .
:juan :workPlace :unileón .

:professor rdfs:label "Professor"@en .
:uniLeón rdfs:label "University of León"@en .
```

A multilingual application can recognize the different components and create a better user experience.

Discussion. Applications can generate more readable information to the end user, especially when they are localized. Decomposing the resources of a dataset to more fine grained atoms can make the dataset more user-friendly.

However, this pattern increases the complexity of the model. It is necessary to find a good balance between verbose models with fine-grained resources and lighter models with a fewer resources and longer descriptions.

See also. This pattern is related to *Link not label*[11] in [18] where the authors describe situations in which it is better to use resources instead of labels.

4.4.2. Lexical information

Description. Using this pattern, we can describe the lexical content of longer descriptions.

Context. Longer descriptions that can not be modified or that are preferred to be kept untouched, can be enriched with lexical information.

Example 11. The following triples describe "University of León" using external descriptions

```
:unileón a lemon:LexicalEntry ;
  lemon:decomposition (
     [ lemon:element :University ]
     [ lemon:element :Of ]
     [ lemon:element :León ]
  );
  rdfs:label "University of León"@en .

:University a lemon:LexicalEntry ;
    lexinfo:partOfSpeech lexinfo:commonNoun ;
    rdfs:label "University"@en ;
    rdfs:label "Universidad"@es .

:Of a lemon:LexicalEntry ;
    lexinfo:partOfSpeech lexinfo:preposition ;
    rdfs:label "of"@en ;
    rdfs:label "de"@es .

:León a lemon:LexicalEntry ;
    lexinfo:partOfSpeech lexinfo:properNoun ;
    rdfs:label "León"
```

[11] http://patterns.dataincubator.org/book/link-not-label.html

Discussion. Providing lexical metadata for a resource supports fully automated software agents. However, this can also add a complexity overhead to the dataset that may be undesired.

See also. There are other ways to describe lexical information. Gananis et al. [27] propose *NaturalOWL* to linguistically annotate ontology concepts while Hellman et al. [31] propose a URI scheme to refer to fragments of a text. In this way, it is possible to annotate text fragment with assertions about their lexical structure.

Metadata attached to textual information can even be applied to support natural language user interfaces. Much research work is being done regarding the generation of natural language descriptions for RDF and for SPARQL querying [23, 44] which can benefit from lexical metadata.

4.4.3. Structured literals

Description. Literals in the RDF model may structured values in XML or HTML. Using structured literals, it is possible to offer longer descriptions leveraging the Internationalization practices that have already been proposed for those languages.

Context. When longer descriptions are obtained from external sources it is better to keep them in their original form or to encode them using XML or HTML. Also, when there are long textual descriptions with lot of internationalization information (localization workflows, multilingualism, etc.) it may be better to employ structured literals following the proposed XML internationalization techniques.

Example 12. A description of the *University of León* could be:

```
:unileón :desc
  "<p>University of
     <span translate="no">León</span>,
     Spain.
  </p>"^^rdf:XMLLiteral .
```

Notice that the example uses the translate attribute from HTML to indicate that *León* should not be translated.

Discussion. Using structured literals makes it possible to leverage existing Internationalization techniques like bi-directionality, ruby annotations, localization notes, etc.

One issue is the interaction between the two abstraction levels: RDF and XML/HTML. Including large portions of structured literals can hinder the linked data approach.

See also. The W3c developed a document on Best Practices for XML internationalization [42] which are complemented by the *Internationalization Tag Set* (ITS) [35]. The new version ITS 2.0 [36] contains categories that are format neutral, supporting both XML, HTML and the RDF based NIF (*NLP Interchange Format*)[12]. It also gives support to localization workflows like those expressed in XLIFF[43].

4.5. Linking

Although the linking process is independent of any natural language, there are two points that must be taken into account when designing multilingual linked data. How to relate resources that refer to the same entity in different languages and how to describe linguistic aspects of a dataset.

4.5.1. Inter-language identity links

Description. Add an owl:sameAs link between two resources that refer to the same entity but contain data in different languages.

Context. When dealing with resources in a multilingual linked data environment, it may be necessary to keep those resources separate and identifiable.

Example 13. Suppose we have information about Armenia in English which is identified by

```
http://hy.example.org#Հայաստան
```

while the URI

```
http://es.example.org#Armenia
```

contains information about Armenia in Spanish. We can declare that both URIs refer to the same thing by asserting:

```
<http://hy.example.org#Հայաստան>
  owl:sameAs
    <http://en.example.org#Armenia> .
```

Discussion. owl:sameAs is a well-known property which is supported by several linked data applications. However, the semantics of owl:sameAs has some implications which may be undesirable. For example, it could be that the information about Armenia in the different languages comes from different sources and thus, contains different data. Using owl:sameAs can then render inconsistencies.

See also. This pattern is a special case of the *Equivalence links* [13] in [18].

4.5.2. Inter-language soft links

Description. Use a soft property to state that two resources are inter-language linked.

Context. owl:sameAs is a very strong property with logical implications and hence must be used careful. Two resources linked by owl:sameAs are supposed to be really the same and reasoners that infer something about one of the resources, will automatically infer the same for the other. A soft link can be either a custom property (i.e. dbo:wikiPageInterLanguageLink in section 5) or a common property such as rdfs:seeAlso, skos:related, etc.

Example 14. Example 13 could be written as follows:

```
<http://hy.example.org#Հայաստան>
  rdfs:seeAlso
    <http://en.example.org#Armenia> .
```

Discussion. Soft links are weaker regarding semantic implications than an owl:sameAs link. Using a custom property (i.e.dbo:wikiPageInterLanguageLink) can provide more freedom but are usually not well recognized by automated software agents. Thus, the use of more common properties with similar semantics (i.e. rdfs:seeAlso, skos:related, etc) should be considered.

See also. Halpin et al. [29] describe the uses and abuses of owl:sameAs and propose to limit its use.

[12] http://nlp2rdf.org/nif-1-0

[13] http://patterns.dataincubator.org/book/equivalence-links.html

4.5.3. Add linguistic metadata

Description. Add linguistic metadata, like localization information or the default language of the dataset.

Context. Some linked data applications, like thesaurus, controlled vocabularies, etc. need to have a finer control on the linguistic terms that they are handling. They may need, for example, to express the linguistic relationship between two concepts or the language in which they are expressed.

Given that it is not possible to have literals as subjects in the RDF model, it is necessary to employ resources as literal representatives and to assert declarations between those resources.

Example 15. The following example shows how we can declare that *Catedrático* means *Professor* and that it is a Spanish term (represented by the code spa in ISO-693-3).

```
:Catedrático
    lexvo:means wordnet:Professor ;
    lexvo:language
      <http://lexvo.org/id/iso639-3/spa> .
```

Discussion. This pattern exposes the semantic relationships between multilingual labels, so they can be connected with other resources.

Using the property lexvo:language it is also possible to declare the language of a dataset, so one can use the *Labels without language tag* pattern (cf. 4.3.3). However, this option is not a standard practice and users may not be aware of those global declarations.

Although it has been proposed to add a property to declare the default language of a named graph in RDF 1.1, it was not accepted. The new JSON-LD working draft allows to declare the default language of a given context.

See also. The Lexvo[14] project [20,21] defines an ontology of linguistic terms. Lexvo proposes a general framework to publish multilingual knowledge bases. For example, it declares a property language and contains URIs for the different languages. In this way, language declarations can be part of the RDF model as well as the relationships between terms.

[14] http://lexvo.org

4.6. Reuse

Reuse is one of the main motivations for linked data. In fact, the best advice to develop linked data solutions is to provide links to existing vocabularies.

It is a good practice to link to popular vocabularies which are well known by the community and can help to integrate different sources of data. In general, the chosen vocabularies can improve the success of a linked data application.

4.6.1. Monolingual vocabularies

Description. Most of the popular vocabularies, such as FOAF or Dublin Core, are not localized at all. The URIs that represent concepts contain English words in ASCII and labels are only provided in English.

Context. Vocabularies with a global scope maintain their terms in a single language, usually English.

Example 16. Most popular vocabularies and ontologies for the semantic web (FOAF, Dublin Core, OWL, RDF Schema, etc.) are monolingual and only employ one language, usually English, both for labels and comments.

Discussion. In monolingual vocabularies, it is easy to control the vocabulary evolution and avoid the appearance of bad translations or ambiguities between language versions. When using monolingual vocabularies in a multilingual application, it is necessary to have a translation layer for those English terms.

Using a monolingual vocabulary as the central knowledge representation system can be constrained by the language used. For example, some languages have different names for one concept that refer to only one name in another language. Also, some concepts in one language might not exactly match to a concept in another language. An example is the concept *Professor*, which in certain regions refers only to an academic holding a chair at a university (e.g. Germany), while it comprises in other regions also secondary or even primary school teachers (such as in Austria). This example illustrates, that such ambiguities can even occur in largely monolingual vocabularies, where the meaning of concepts differs in various regions.

See also. This pattern is opposed to the *Multilingual vocabularies* pattern 4.6.2. Most of these vocabularies use *descriptive URIs* (4.1.1) with English terms.

4.6.2. Multilingual vocabularies

Description. Define vocabularies and ontologies where the concepts contain translations for several languages.

Context. In a multilingual linked data application where we want to have more control about the translation process, it is better to provide our own translations defining multilingual versions of the ontologies.

Example 17. In our running example, we can define a multilingual vocabulary for university positions with declarations like:

```
:position a owl:DatatypeProperty ;
  rdfs:domain :UniversityStaff ;
  rdfs:label "Position"@en ;
  rdfs:label "Puesto"@es .

:UniversityStaff a owl:Class ;
  rdfs:label "University staff"@en ;
  rdfs:label "Trabajador universitario"@es .
```

Discussion. Multilingual vocabularies offer an elegant solution for applications that need to express information in local languages. There is no need to translate labels and comments and the users of those languages can have access to more standard textual representations in their languages.

Some common vocabularies use only one language, usually English, as a canonical textual representation of the different concepts. Some concepts are difficult to translate and there may appear ambiguities in the translations. For example, the label *Professor* may be translated to *Profesor* in Spanish. However, the meaning of those concepts is different (in Spanish it is usually preferred as *Catedrático*).

See also. There are a number of multilingual vocabularies, like Agrovoc[15] or Eurovoc[16]. This pattern is opposed to the *Monolingual vocabularies* pattern 4.6.1. In Hyland et al. [33] it is proposed as a quality selection criteria the use of vocabularies that contain descriptions in more than one language.

[15] http://aims.fao.org/standards/agrovoc/about

[16] http://eurovoc.europa.eu/

4.6.3. Localize existing vocabularies

Description. Enrich existing vocabularies with local translations.

Context. Localized applications that need to represent information in their local languages from external vocabularies that are not localized can define their own translations.

Example 18. A linked data application in Spanish may use the Dublin Core vocabulary to indicate the contributors of a given work. The end-user should see the labels in his own language. To that end, one can add a localized label to dc:contributor as:

```
dc:contributor rdfs:label "Colaborador"@es .
```

Discussion. A multilingual linked data application could transparently select the tagged literal corresponding to dc:contributor in its preferred language. Polluting well known vocabularies with localized literals may be controversial and should be handled with caution.

See also. This pattern follows the *Anyone-can-say-Anything-about-Any-topic* (AAA) paradigm [2]. This pattern can be considered a special case of the *Annotation*[17] pattern from [18] which says that *it is entirely consistent with the Linked Data principles to make statements about third-party resources.*

4.6.4. New localized vocabularies

Description. This pattern advocates to create new localized properties and classes and relate them to existing ones using the owl:sameAs, owl:equivalentProperty or owl:equivalentClass properties.

Context. Linked data applications that need localized versions of existing vocabularies but prefer to keep the original vocabularies untouched.

Example 19. One can create a custom :colaborador property and then state that this property is equivalent to dc:contributor.

```
dc:contributor
    owl:equivalentProperty :colaborador .
:colaborador
    rdfs:label "Colaborador"@es .
```

[17] http://patterns.dataincubator.org/book/annotation.html

Discussion. This pattern gives freedom to vocabulary creators to tailor the vocabulary according to their exact needs. However, it can be more difficult for both humans and software agents to recognize and consume these new properties and classes.

See also. This pattern is related to the *Equivalence links*[18] pattern from [18] and to the *Link base*[19] pattern which proposes to partition the core data from the links.

5. Use case: DBpedia Internationalization

DBpedia is an effort to extract structured information from Wikipedia and publish this information as Linked Open Data [5]. Due to the interdisciplinary nature and broad term coverage of Wikipedia, DBpedia has managed to became one of the main hubs of LOD cloud[20]. The early versions of the DBpedia used only the English Wikipedia as its sole source since it is the most comprehensive language edition. However, there exists much knowledge in other Wikipedia language editions that was not made available by DBpedia.

There have been three main attempts to also harvest this multilingual knowledge:

1. by DBpedia, extracting simple types of multilingual information,
2. by [7], extending the first approach with the use of IRIs and
3. by [34], extending the use of IRIs with a dereferencing solution and the addition of links between different DBpedia language editions.

The DBpedia project is an excellent case study for a multilingual dataset. In the following paragraphs we will show how the multilingual linked data patterns employed throughout the internationalization of the DBpedia project.

A common practice of the DBpedia project is the use of declarative URIs. For every page in Wikipedia, i.e. http://en.wikipedia.org/wiki/Armenia, a resource is created in the form http://dbpedia.org/resource/Armenia.

Instead of opaque URIs (cf. 4.1.2) DBpedia employes descriptive URIs, firstly restricted to ASCII characters and percent encoding extended characters (cf. 4.1.1).

Among other information, DBpedia added a label extracted from the page title using a language tag (cf. 4.3.2).

Example 20. The data about Armenia in English DBPedia is:

```
dbp−en:Armenia rdfs:label "Armenia"@en ;
    ...
```

To obtain information about the same topic in other Wikipedia language editions, DBpedia uses the Wikipedia *Inter-Language Links* (ILL)[21]. ILLs are special links in a Wikipedia page that link to the exact or meaningwise closest article in a different Wikipedia language edition.

Example 21. http://en.wikipedia.org/wiki/Armenia has the following ILLs:

```
http://es.wikipedia.org/wiki/Armenia
http://hy.wikipedia.org/wiki/Հայաստան
```

Early versions of DBpedia extracted only the English article and discarded this type of information. Later on, DBpedia extracted data from other Wikipedia language editions but, if and only if the non-English article had an ILL to an English article. If it did not, the article was discarded, otherwise it would use the English URI as an identifier.

Example 22. All the aforementioned Wikipedia articles would have http://dbpedia.org/resource/Armenia as their URI and use language tagged labels for translations (cf. 4.3.2).

```
dbp−en:Armenia rdfs:label "Armenia"@en ;
dbp−en:Armenia rdfs:label "Հայաստան"@hy ;
dbp−en:Armenia rdfs:label "Armenia"@es ;
```

This approach helped the DBpedia project to enrich the existing data with multilingual information but had the following drawbacks:

[18] http://patterns.dataincubator.org/book/equivalence-links.html

[19] http://patterns.dataincubator.org/book/link-base.html

[20] Linking Open Data cloud diagram, by Richard Cyganiak and Anja Jentzsch. http://lod-cloud.net/

[21] http://en.wikipedia.org/wiki/Help:Interlanguage_links

1. ILLs are not always exact translations thus, a (small) part of this enrichment was inaccurate,
2. even exact translations may have conflicting information due to stalled content or different views on the subject (i.e. the population of a town) thus, collapsing everything to the same namespace (http://dbpedia.org/resource/) could produce problems and
3. local articles without an English translation were discarded.

The solution that addressed these issues was the use of separate namespaces for every language (cf. 4.1.5), following the Wikipedia naming pattern.

Example 23. The Armenian Wikipedia article, http://hy.wikipedia.org/wiki/Հայաստան would produce the following identifier:

```
http://hy.dbpedia.org/resource/Հայաստան
```

Since DBpedia was using descriptive identifiers, non-Latin languages used Internationalized local names (cf. 4.1.4) to ease the manual SPARQL querying and visual resource identification.

Example 24. Different language versions of Armenia in DBPedia:

```
dbp−en:Armenia rdfs:label "Armenia"@en
dbp−es:Armenia rdfs:label "Armenia"@es
dbp−hy:Հայաստան rdfs:label "Հայաստան"@en
```

In order to keep the ILLs, DBpedia introduced a new predicate in the DBpedia ontology, http://dbpedia.org/ontology/wikiPageInterLanguageLink, and produced a triple for every ILL.

By post-processing the ILLs it was proven that when two articles had both an ILL to each-other (two-way ILL) it was relatively safe to assume an exact translation. Thus, ILLs were transformed to *owl:sameAs* in the case of two-way links or *rdfs:seeAlso* in the case of one-way links (cf. 4.5.1 and 4.5.2).

Example 25. ILLs between the different resources that represent Armenia in the different datasets:

```
dbp−en:Armenia dbp:int−link dbp−hy:Հայաստան
dbp−hy:Հայաստան dbp:int−link dbp−en:Armenia
dbp−es:Armenia dbp:int−link dbp−en:Armenia
dbp−en:Armenia owl:sameAs dbp−hy:Հայաստան
dbp−hy:Հայաստան owl:sameAs dbp−en:Armenia
dbp−es:Armenia rdfs:seeAlso dbp−en:Armenia
```

6. Related work

This paper can be seen as a continuation of the linked open data patterns book by Leigh Dodds and Ian Davis [18]. Although it contains some patterns related to multilingualism, the book is about linked data in general, while in this paper we concentrated on patterns that are crucial for multilingualism. Along the presentation of our catalog patterns, we aligned the our catalog patterns with the ones in that book.

There are also a good number of best practices and guidelines about publishing linked open data [1,12,33].

The linguistic community also considered challenges of a multilingual web of data [28]. The benefits of interlinked linguistic resources are presented in [6].

A lot of work has been done regarding multilingual ontologies. In particular, the *Monnet*[22] project has been devoted to the development of cross-lingual knowledge representation and extraction using ontologies. As a use case, [25] describes a system to semi-automate the localization of ontologies while [39] proposes the *Language Information Repository* model and describes three ways to model multilinguality in ontologies: include multilingual data in the ontology metamodel, combine the meta-model with a mapping model and associate the meta-model with a multilingual linguistic model.

[16] describes the limitations of the label systems in RDF, SKOS and OWL. The authors propose the *LexInfo* model which allows authors to associate linguistic information with the different elements of an ontology. McCrae et al. [37] propose the *lemon* model which consists of a lexicon object with a number of lexical entities. LexInfo was designed as a model for associating linguistic information with ontologies and provides special data categories for this mapping. LexInfo was originally based on LMF, however, this was found to be difficult to work with in a linked data setting, and the lemon model was created, as an adaptation of LMF to the challenges of linked data. Thus, LexInfo 2.0 acts as a data category repository and ontological model of lemon data.

[22] http://www.monnet-project.eu

DBpedia can be viewed as a comprehensive resource for linguistic research. Several authors have already used DBpedia to improve NLP tools [38].

7. Conclusions and future work

The web of data is not just for machines. At the end, applications based on the web of data will be used by humans and human beings speak many different languages.

We have proposed a set of guidelines or patterns to be taken into account when developing multilingual linked data applications. Some of those guidelines are frequently used while others, like language content negotiation, are rarely used. Future work should be done to really assess the benefits of the different approaches.

There are other issues like Unicode support by semantic web tools and standards, language declarations in Microdata, that are currently the subject of working groups. In fact, during the writing of this paper the RDF WG is working in RDF 1.1. Although we were following the discussions, in this paper we did not cover aspects that could change in future; so the patterns presented herein should not be affected by future changes.

Other internationalization topics like text direction, ruby annotations, notes for localizers, translation rules, etc. are handled by the W3C Internationalization group and the ITS 2 effort. It is expected that some alignment between that work and multilingual linked data will provide very fruitful results in the future which could result in the appearance of new patterns for the localization of linked data.

Recently there has also been some work on measuring the quality of linked open data. Hogan et al. [32] takes 14 principles for naming, linking, describing and referencing resources into account. Those principles are not focused on multilingual aspects, although a similar study can be carried out considering the patterns described in this paper. Increasing the quality of linked data is a very important goal, that must take into account the multilingual nature of people.

8. Acknowledgments

This work has been partially funded by Spanish project MICINN-12-TIN2011-27871 ROCAS (Reasoning on the Cloud by Applying Semantics). Some of these best practices were presented at the *Multilingual Web Conference*, Dublin 2012 and at the *Multilingual Linked Open Data for Enterprises Workshop*, Leipzig, 2012, we appreciate the comments and discussions of the participants of those events. We would also like to acknowledge the comments given by: Jose María Álvarez Rodríguez, Basil Ell, Manuel T. Carrasco, Aidan Hogan, Richard Cyganiak, Mohamed Morsey, John McCrae, Pablo Mendes, Elena Montiel and Jeni Tennison.

References

[1] Best practice recipes for publishing RDF vocabularies. W3c Wordking Drafft, 2008.

[2] *Semantic Web for the Working Ontologist, Effective Modeling in RDFS and OWL*. Morgan Kaufmann, 2 edition, 2011.

[3] Character model for the world wide web 1.0: Normalization. W3c Wordking Drafft, 2012.

[4] Turtle, terse rdf triple language. World Wide Web Consortium, Working Draft, WD-Turtle, July 2012.

[5] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives. DBpedia: A nucleus for a web of open data. In *Proceedings of the 6th International Semantic Web Conference (ISWC)*, volume 4825 of *Lecture Notes in Computer Science*, pages 722–735. Springer, 2008.

[6] S. Auer and S. Hellmann. The web of data: Decentralized, collaborative, interlinked and interoperable. In *8th International Conference on Language Resources and Evaluation*, 2012.

[7] S. Auer, M. Weidl, J. Lehmann, A. J. Zaveri, and K.-S. Choi. I18n of semantic web applications. In P. F. Patel-Schneider, Y. Pan, P. Hitzler, P. Mika, L. Z. 0007, J. Z. Pan, I. Horrocks, and B. Glimm, editors, *The Semantic Web - ISWC 2010 - 9th International Semantic Web Conference, ISWC 2010, Shanghai, China, November 7-11, 2010, Revised Selected Papers, Part II*, volume 6497 of *Lecture Notes in Computer Science*, pages 1–16. Springer, 2010.

[8] T. Berners-Lee. Univeral resource identifiers – axioms of web architecture. http://www.w3.org/DesignIssues/Axioms.html, 1996.

[9] T. Berners-Lee. Cool uris don't change. http://www.w3.org/Provider/Style/URI.html, 1998.

[10] T. Berners-Lee. Linked data - design issues. http://www.w3.org/DesignIssues/LinkedData.html, July 2006.

[11] T. Berners-Lee, R. Fielding, and L. Masinter. RFC 3986, uniform resource identifier (URI): Generic syntax. Request For Comments (RFC), 2005.

[12] C. Bizer, R. Cyganiak, and T. Heath. How to publish Linked Data on the Web. http://www4.wiwiss.fu-berlin.de/bizer/pub/LinkedDataTutorial/, Oct. 2008.

[13] C. Bizer, T. Heath, and T. Berners-Lee. Linked data - the story so far. *International Journal on Semantic Web and Information Systems*, pages 1–26, Mar 2010.

[14] C. Caracciolo, A. Stellato, A. Morshed, G. Johannsen, S. Rajbhandari, Y. Jacques, and J. Keizer. Thesaurus maintenance, alignment and publication as linked data - the agrovoc use case. *International Journal of Metadata, Semantics and Ontologies*, 7(1):65–75, 2012.

[15] F. Cifuentes-Silva, C. Sifaqui, and J. E. L. Gayo. Towards an architecture and adoption process for linked data technologies in open government contexts (a case of study for the library of congress of chile). 2011.

[16] P. Cimiano, P. Buitelaar, J. McCrae, and M. Sintek. Lexinfo: A declarative model for the lexicon-ontology interface. *Web Semantics: Science, Services and Agents on the World Wide Web*, 9(1), 2011.

[17] R. Cyganiak. Semweb rules of thumb. `http://www.w3.org/wiki/User:Rcygania2/RulesOfThumb`.

[18] I. Davis and L. Dodds. *Linked Data Patterns*. 2010.

[19] M. Davis and M. Suignard. Unicode security considerations. Technical Report 36, Unicode Technical Report, 2012.

[20] G. de Melo and G. Weikum. Language as a foundation of the Semantic Web. In C. Bizer and A. Joshi, editors, *Proceedings of the Poster and Demonstration Session at the 7th International Semantic Web Conference (ISWC 2008)*, volume 401 of *CEUR WS*, Karlsruhe, Germany, 2008. CEUR.

[21] G. de Melo and G. Weikum. Towards universal multilingual knowledge bases. In P. Bhattacharyya, C. Fellbaum, and P. Vossen, editors, *Principles, Construction, and Applications of Multilingual Wordnets. Proceedings of the 5th Global WordNet Conference (GWC 2010)*, pages 149–156, New Delhi, India, 2010. Narosa Publishing.

[22] M. Dürst and M. Suignard. Internationalized resource identifiers. Technical Report 3987, IETF, 2005.

[23] B. Ell, D. Vrandecic, and E. Simperl. Spartiqulation: Verbalizing sparql queries. In *Proceedings of the International Workshop on Interacting with Linked Data (ILD 2012), Extended Semantic Web Conference (ESWC)*. CEUR-WS.org, Mai 2012.

[24] B. Ell, D. Vrandečic, and E. Simperl. Labels in the web of data. In *Proceedings of the 10th international conference on The semantic web - Volume Part I*, ISWC'11, pages 162–176, Berlin, Heidelberg, 2011. Springer-Verlag.

[25] M. Espinoza, A. Gómez-Pérez, and E. Mena. Enriching an ontology with multilingual information. In *Proceedings of the 5th European semantic web conference on The semantic web: research and applications*, ESWC'08, pages 333–347, Berlin, Heidelberg, 2008. Springer-Verlag.

[26] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. Hypertext transfer protocol – http/1.1 (rfc 2616). RFC 2616 (Proposed Standard), 1999.

[27] D. Galanis and I. Androutsopoulos. Generating multilingual descriptions from linguistically annotated owl ontologies: the NaturalOWL system. In *11th European Workshop on Natural Language Generation*, 2007.

[28] J. Gracia, E. Montiel-Ponsoda, P. Cimiano, A. Gómez-Pérez, P. Buitelaar, and J. McCrae. Challenges for the multilingual web of data. *Web Semant.*, 11:63–71, Mar. 2012.

[29] H. Halpin, P. J. Hayes, J. P. McCusker, D. L. McGuinness, and H. S. Thompson. When owl: sameas isn't the same: an analysis of identity in linked data. In *Proceedings of the 9th international semantic web conference on The semantic web - Volume Part I*, ISWC'10, pages 305–320, Berlin, Heidelberg, 2010. Springer-Verlag.

[30] T. Heath and C. Bizer. *Linked Data: Evolving the Web into a Global Data Space*. Synthesis Lectures on the Semantic Web: Theory and Technology. Morgan and Claypool, 2011.

[31] S. Hellmann, J. Lehmann, and S. Auer. Towards an Ontology for Representing Strings for the NLP Interchange Format (NIF). In *EKAW*, 2012.

[32] A. Hogan, J. Umbrich, A. Harth, R. Cyganiak, A. Polleres, and S. Decker. An empirical survey of linked data conformance. *Web Semantics: Science, Services and Agents on the World Wide Web*, Apr. 2012.

[33] B. Hyland, B. Villazón-Terrazas, and M. Hausenblas. Best practices for publishing linked data. `http://www.w3.org/TR/gld-bp/`, April 2012.

[34] D. Kontokostas, C. Bratsas, S. Auer, S. Hellmann, I. Antoniou, and G. Metakides. Internationalization of linked data: The case of the greek dbpedia edition. *Web Semantics: Science, Services and Agents on the World Wide Web*, (0), 2012.

[35] C. Lieske and F. Sasaki. Internationalization tag set (its) version 1.0. `http://www.w3.org/TR/its/`, April 2008.

[36] S. McCane, D. Lewis, A. Lommel, J. Kosek, F. Sasaki, and Y. Savourel. Internationalization tag set (its) version 2.0. `http://www.w3.org/TR/its20/`, December 2012.

[37] J. McCrae, D. Spohr, and P. Cimiano. Linking lexical resources and ontologies on the semantic web with lemon. In *Proceedings of the 8th extended semantic web conference on The semantic web: research and applications - Volume Part I*, ESWC'11, pages 245–259, Berlin, Heidelberg, 2011. Springer-Verlag.

[38] P. N. Mendes, M. Jakob, and C. Bizer. Dbpedia for nlp: A multilingual cross-domain knowledge base. In *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, Istanbul, Turkey, May 2012.

[39] E. Montiel-Ponsoda, G. A. de Cea, A. Gómez-Pérez, and W. Peters. Enriching ontologies with multilingual information. *Natural Language Engineering*, 17(3):283–309, 2011.

[40] E. Montiel-Ponsoda, D. Vila-Suero, B. Villazón-Terrazas, G. Dunsire, E. Escolano, and A. Gómez-Pérez. Style guidelines for naming and labeling ontologies in the multilingual web, 2011.

[41] A. Phillips and M. Davis. Tags for Identifying Languages. Technical Report 47, Internet Engineering Task Force, September 2009.

[42] Y. Savourel, J. Kosek, and R. Ishida. Best practices for xml internationalization. `http://www.w3.org/TR/`

xml-i18n-bp/, February 2008.

[43] Y. Savourel, J. Reid, T. Jewtushenko, and R. M. Raya. Xml localization interchange file format (XLIFF). OASIS Standard, 1999.

[44] S. Shekarpour, S. Auer, A. Ngonga, D. Gerber, S. Hellmann, and C. Stadler. Keyword-driven sparql query generation leveraging background knowledge. In *International Conference on Web Intelligence*, 2011.

[45] T. Takada. Multilingual information exchange through the world-wide web. In *First WWW Conference*, 1994. Online, accessed 31-oct-2012.

[46] P. Vossen, editor. *EuroWordNet: a multilingual database with lexical semantic networks*. Kluwer Academic Publishers, Norwell, MA, USA, 1998.