Weaving the Pedantic Web

Title	Weaving the Pedantic Web
Author(s)	Hogan, Aidan;Harth, Andreas;Passant, Alexandre;Decker, Stefan;Polleres, Axel
Publication Date	2010
Publisher	CEUR

Weaving the Pedantic Web*

Aidan Hogan[†], Andreas Harth[†‡], Alexandre Passant[†], Stefan Decker[†], Axel Polleres[†]

[†]Digital Enterprise Research Institute, National University of Ireland, Galway
[‡]AIFB, Karlsruhe Institute of Technology, Germany

[†]{firstname.lastname}@deri.org, [‡]harth@kit.edu

ABSTRACT

Over a decade after RDF has been published as a W3C recommendation, publishing open and machine-readable content on the Web has recently received a lot more attention, including from corporate and governmental bodies; notably thanks to the Linked Open Data community, there now exists a rich vein of heterogeneous RDF data published on the Web (the so-called "Web of Data") accessible to all. However, RDF publishers are prone to making errors which compromise the effectiveness of applications leveraging the resulting data. In this paper, we discuss common errors in RDF publishing, their consequences for applications, along with possible publisher-oriented approaches to improve the quality of structured, machine-readable and open data on the Web.

1. INTRODUCTION

Based on the simple principle of using URIs to name and link *things* – not just documents – the Resource Description Framework (RDF) offers a standardised means of representing information on the Web such that: (i) structured data is available to all over the Web; (ii) data can be handled through standard APIs and applications; (iii) the meaning of the data is well-defined using lightweight ontologies (or vocabularies); and (iv) data is interoperable with other RDF on the Web and can be re-used and extended by other publishers and application developers.

Over the past few years, many Web publishers have turned to RDF as a means of disseminating information in an open and machine-interpretable way, resulting in a "Web of Data" which now includes interlinked content exported from corporate bodies (e.g., BBC, New York Times, Freebase), community efforts (e.g., Wikipedia, GeoNames), biomedical datasets (e.g., DrugBank, Linked Clinical Trials) – even UK governmental entities, where public sector organisations must now additionally disclose their consultations in RDF[1]. Applications and search engines are now starting to exploit this rich vein of structured and linked data [9].

For example, Figure 1 shows the results returned by the VisiNav (`http://visinav.deri.org/`) system for the query "show me all American female models who have also won an Academy Award for Best Supporting Actress", constructed using facets in the user-interface; the results are automatically aggregated from thirteen distinct sources.

However, all has not been plain sailing: this new paradigm in Web publishing and interaction [11] has inevitably led to many teething problems. As we will discuss in this paper, there exists a lot of noise within the Web of Data which inhibits applications from effectively exploiting this rich lode of open, well-defined and structured information.

To illustrate, we introduce Alice: a hypothetical end-user of an application for searching and browsing the Web of Data. Alice loads some interesting data about herself and is immediately impressed by the integrated view of data from publication, blog, social network and workplace exporters; however, for every second resource she explores, the application cannot locate or parse any relevant data. She tries to load her publications into a calendar view, but one quarter of them are missing as the dates/times contain illegal values. She wants more information relating to properties and classes used to describe herself, but some do not exist; discouraged, she clicks on a friend of hers but finds that he has 1,169 names and email addresses (she knew him as "Bob"). She begins to notice that all resources she explores are instances of nine strange properties – and then the final straw: she now finds out that her professor is actually a document.

We will provide evidence in this paper as to how Alice could have had such an experience browsing the Web of Data. In so doing, we will take stock of some of the difficulties currently apparent in RDF publishing, and discuss how we – and the now decade old Semantic Web community at large – can help to improve the current and future quality of RDF data published on the Web.

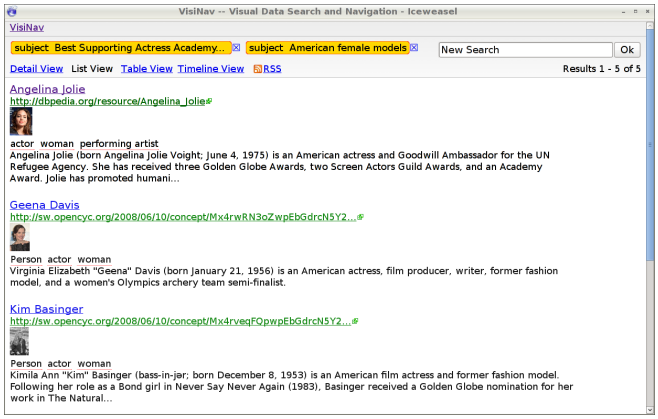

Figure 1: Results from VisiNav showing American models who have won Best Supporting Actress in the Academy Awards.

*We would like to acknowledge and thank Richard Cyganiak, Michael Hausenblas, Stphane Corlosquet and Antoine Zimmermann with whom we co-founded the Pedantic Web Group. We would also like to thank anonymous reviewers of various incarnations of this paper for their valued feedback. The work presented in this paper has been funded in part by Science Foundation Ireland under Grant No. SFI/08/CE/I1380 (Lion-2) and by an IRCSET Postgraduate scholarship.

[1]`http://coi.gov.uk/guidance.php?page=315`

LDOW2010, April 27, 2010, Raleigh, USA.

2. WEB OF DATA ANALYSIS

We herein present some analysis based on an RDF dataset retrieved from the Web in April 2009 using MultiCrawler [8]. We performed a seven-hop breadth-first crawl for RDF/XML documents where we enforced a maximum of 5,000 crawled documents per pay-level-domain (or PLD, viz.: a domain that requires payment, such as `deri.ie` or `data.gov.uk`) so as to ensure a diverse Web dataset covering a wide spectrum of publishers. Indeed, we only crawled for RDF/XML and not for other formats such as RDFa; RDF/XML is currently by far the most popular format with RDFa growing in popularity. Still, one could expect a small percentage of documents to contain – e.g., RDFa metadata – which we admittedly overlook in our illustrative statistics.

The crawl accessed 149,057 URIs (including 39,439 redirects), 54,836 (36.8%) of which resulted in valid RDF/XML documents (almost precisely 50% excluding redirects). The final dataset contains 12,534,481 RDF statements mentioning 1,598,521 URIs – including 5,850 classes and 9,507 properties.

Based on this dataset, we present selected issues in RDF data published on the Web. We focus on errors that we can systematically detect, and thus one should not consider the following an exhaustive list; similarly, it is important to note – given the diminutive scale and perhaps even age of our dataset – that the statistics presented herein are intended to be illustrative, not exhaustive. That said, we still claim that the analysis of our dataset offers a valuable insight into current issues relating to RDF Web publishing: although interpolating the *exact* prevalence of such problems to the entire Web of Data may not be sensible, our statistics should offer an indication as to the *relative* and *approximate* prevalence of such problems.

Throughout the paper, we endeavour to present examples of the various publishing errors by giving links to RDF Web documents exhibiting such. Note that the purpose of providing these examples is to: (i) to give concrete and tangible examples to the errors, giving indications as to how they might have occurred, how they might be presently solved, and how they could be avoided in future; (ii) show that noise is present in a diverse range of sources, describing a diverse range of domains; and (iii) to show that errors in RDF publishing are not only the result of inexperience – we show examples of errors in academic publishing, community-based publishing, popular vocabularies, and even documents published by the authors of this paper. The purpose of the examples is thus not to "point the finger", but to give an honest appraisal of such issues so as to identify possible directions forward.

For posterity, we provide snapshots of documents and enumerate the namespace prefixes referenced in this paper at `http://aidanhogan.com/pedantic/`.

In order to structure the highlighted issues, we identify four categories of symptoms:

- **incomplete**: equatable to a dead-link in the current HTML web – a software agent will not be able to retrieve data relevant to a particular task;
- **incoherent**: a software agent will not be able to correctly interpret some local piece of data as either the publisher or vocabulary maintainer would expect;
- **hijack**: a software agent will not be able to correctly interpret some remote piece of data as would be expected;
- **inconsistent**: a software agent will interpret a contradiction in the data.

Finally, we will also endeavour to provide discussion for each issue, both from the perspective of publishers and from the perspective of data consumers.

We begin with issues relating to how data is found and accessed; then discuss parsing and syntax issues; look at reasoning issues, including inconsistent data; and finally, introduce and discuss *ontology hijacking*.

2.1 URI/HTTP: accessibility and derefencability

As previously alluded to, the Linked Open Data movement has been integral to RDF publishing on the Web, emphasising four basic principles [2]: (i) use URIs as names for things; (ii) use HTTP URIs so that those names can be looked up; (iii) provide useful information when a look-up on that URI is made; and (iv) include links using external URIs.

With regards to providing information about a resource upon a HTTP lookup of its URI – called dereferencing – emphasis is placed on providing information in RDF and disambiguating identification of information resources (document URIs) from non-information resources (entities described in those documents). Now, using statistics of our crawl which consisted of lookups on URIs in the data, we can draw some initial conclusions relating to Linked Data practices on the Web of Data.

2.1.1 Dereferencability issues

Category: **incomplete**

In accordance with "use HTTP URIs so that those names can be looked up", dereferencing a URI consists of retrieving content as defined by RFC3986[2].

Firstly, 5.3% of URIs returned an error (`4xx` client error/`5xx` server error) response code, in conflict with the third Linked Data principle above: "provide useful information when a look-up on that URI is made". In most such – admittedly relatively rare – cases simply nothing exists at that location and a `404 Not Found` code is returned (4.3% overall, 81% of error codes).

Secondly, 26.5% of URI lookups resulted in a redirect (`30x` code). In fact, Linked Data principles encourage the use of redirects, particularly for identifying non-information resources (i.e. URIs which denote things rather than files): specifically, the `303` redirect is recommended. Of the redirection URIs, 55.1% (14.6% of total) offered a `303` redirect to another location as recommended; however, 30.2% (8% of total) used a `302` redirect and the remaining 14.7% (3.9% of total) used a `301` redirect.

In the machine-oriented world of Linked Data, publishers should be even more careful to avoid broken links and to make URIs dereferencable, thus enabling automatic data-access for Semantic Web applications and providing them – and ultimately end-users – a complete, coherent picture.

Publisher Recommendations: Publishers should carefully follow Linked Data best practices when "minting" URIs.

Consumer Recommendations: Applications should not expect high recall when dereferencing URIs found in RDF data: for high recall, applications may have to consider prefetching/data-warehousing approaches.

2.1.2 No structured data available

Category: **incomplete**

Excluding redirects, 92.8% of URIs return a `200 OK` response code along with content; but what do these documents contain? Linked Data principles require that useful

[2]`http://labs.apache.org/webarch/uri/rfc/rfc3986.html`

data be returned upon lookup of a URI from the Web of Data, with particular emphasis on returning RDF. Thus, from our crawl requesting `application/rdf+xml` content, we would reasonably expect a high percentage of documents returning RDF/XML[3].

Of the 101,709 URIs which returned content with response code `200 OK`, we observed that only 45.4% of URIs report a content-type `application/rdf+xml`, with a further 34.8% reporting `text/html`. Commonly in RDF data, information resource URIs are used to identify themselves (or more problematically to identify related resources); for example, in RDF, HTML documents are naturally identified using their native URI. In almost all instances of a non-RDF content-type, the URI is simply a document without any supporting RDF metadata. Hence, as before, Semantic Web agents will not be able to properly exploit the content as expected by end-users.

Publisher Recommendations: HTML pages – especially those whose URIs are mentioned in RDF documents – could be embedded with RDFa.

Consumer Recommendations: A possible – and admittedly quick and dirty – solution to avoid dead-links would be to convert the header information of HTTP URIs into RDF using the terms from the W3C published "HTTP Vocabulary in RDF 1.0"[4]. More ambitiously, a system may consider extracting RDF from non-RDF content, such as the title of a HTML page or metadata for images. Such measures would ensure that at the very least, *some* structured information can be retrieved for a wider variety of URIs, thus avoiding 'dead-links'.

2.1.3 Misreported content-types

Category: **incomplete**

A HTTP response contains an optional header field stating the content type of the returned file. A consumer application can then decide from the header whether the content is suitable for consumption, and whether the content should be accessed. However, we observed that RDF/XML content is commonly returned with a reported content-type other than `application/rdf+xml`: from our crawl, 16.9% of valid RDF/XML documents were returned with an incompatible or more generic content type; e.g.,: `text/xml` (9.5%), `application/xml` (5.9%), `text/plain` (1%) & `text/html` (0.4%).

Publisher Recommendations: Publishers should ensure that the most specific available MIME-type is reported for their content.

Consumer Recommendations: Herein, a trade-off exists for consumer agents: an agent with emphasis on performance may still use the reported content-type to filter non-supported content formats, whereas an agent with more emphasis on recall should relax – or possibly ignore – filtering based on reported content-type.

2.2 Syntax errors

2.2.1 RDF/XML Syntax Errors

Category: **incomplete**

At the outset of the Semantic Web movement, publishers opted to employ the existing XML standard to encode RDF; RDF/XML is still the most popular means of publishing RDF today. Although its syntax is quite complex, we encountered relatively few syntax errors in RDF/XML documents accessed during our crawl. Of the 46,136 documents which return response code `200` and content-type `application/rdf+xml`, only 571 (1.2%) were invalid RDF/XML documents; usually caused by simple errors such as unescaped special characters, misuse of RDF/XML shortcuts, and omission of namespace. Again, such issues are relatively rare, presumably due to use of mature RDF/XML APIs for producing data and the popularity of the W3C RDF/XML validation service[5].

Publisher Recommendations: Publishers should use an appropriate syntactic validator for their content, or only use trusted APIs to produce content.

Consumer Recommendations: Applications could possibly investigate the use of tools for fixing syntax errors: e.g., use standard XML syntax cleaning tools for XML-based RDF syntax. We have no experience in using such tools, and they would have to be evaluated in the given application scenario: again in any case, syntax errors are admittedly rare and such concerns would only apply to applications with a large emphasis on high recall.

2.3 Reasoning: noise and inconsistency

Thus far, we have seen that about half of the URIs used to identify resources in the Web of Data resolve to some valid RDF/XML data. We now look at issues relating to the data contained within those documents: i.e., what they say and how the machine interprets the data.

Layered on top of RDF are the core RDF Schema (RDFS) and Web Ontology Language (OWL) standards, which allow for defining the semantics or *meaning* of RDF data through definitions of *classes* and *properties* in schemas/ontologies. For example, the Friend Of A Friend (FOAF)[6] project publishes OWL definitions of a set of classes and properties which forms a structured and popular vocabulary for describing people in RDF.

Classes represent a grouping of resources: e.g., FOAF defines the class `foaf:Person` and one can assign `ex:Alice` and `ex:Bob` as members of this class. Using RDFS and OWL, a publisher can then define characteristics of such classes (and, thus, of all of its members); e.g., by defining `foaf:Person` as a subclass of `foaf:Agent`, FOAF implies that `ex:Alice`, `ex:Bob` and all other `foaf:Persons` are also members of the class `foaf:Agent`.

Properties represent the definable attributes of resources, and also relationships that are possible between resources; e.g., FOAF defines `foaf:knows` as a relationship that can exist from one member of `foaf:Person` to another, or that members of `foaf:Person` can have the attribute `foaf:surname` which has a string value. Other publishers across the Web can then reuse and extend definitions of classes and properties – such as the ones from FOAF.

Thereafter, *reasoning* can use the semantics of these classes and properties to *interpret* the data, and to *infer* new knowledge (e.g., that `ex:Alice` is also a `foaf:Agent`, or that if `ex:Alice foaf:knows ex:Bob`, then `ex:Alice` and `ex:Bob` are `foaf:Persons`).

Some errors in RDF only reveal themselves after reasoning – e.g., some unforeseen incorrect inferences occur – and as such, can stay hidden from the publisher. In this section, we will look at issues relating to the interpretation of RDF data on the Web – in particular focussing on reasoning issues; in order to shed light on such issues, we applied reasoning over our crawl using the Scalable Authoritative OWL Reasoner (SAOR) [12], which we will discuss as pertinent.

2.3.1 Atypical use of collections, containers and reification

[3]

[4] `http://www.w3.org/TR/HTTP-in-RDF10/`

[5] `http://www.w3.org/RDF/Validator/`

[6] `http://foaf-project.org/`

Category: **incoherent**

There is a set of URI names which are reserved by the RDF specification for special interpretation in a set of triples; although the RDF specification does not formally restrict usage of these reserved names, misuse is often inadvertent.

We firstly discuss the RDF collection vocabulary, which consists of four constructs: {`List`, `first`, `rest`, `nil`}. Indeed, few examples of atypical collection usage exist on the Web, probably attributable to widespread usage of the RDF/XML shortcut `rdf:parseType="Collection"` for specifying collections; this shortcut shields users from the underlying complexity of collections on the triple level and generally ensures typical collection use. The only atypical collection usage we found in our Web-crawl was one document which specified resources of type `List` without `first` or `rest` properties attached[7].

A related issue is that of atypical container usage, which is concerned with the following constructs: `Alt`, `Bag`, `Seq`, `_1`...`_n` and the syntactic keyword `li`. Again, atypical container usage is uncommon on the Web: we found one domain (viz. `semanticweb.org`[8]) which, in 229 documents, exports RDF containers without choosing a type of `Alt`, `Bag` or `Seq`.

Finally, there may exist atypical usage of the reification constructs: `Statement`, `subject`, `predicate`, `object`. However, in our dataset we only found one such example[9] wherein `predicate` is assigned a blank node value and used alone without `subject` or `object`.

Publisher Recommendations: Where possible, publishers should abide by the standard usage of such RDF terms to enable interoperability.

Consumer Recommendations: Although we found that atypical usage of the core RDF terms is relatively uncommon, consumer applications should be tolerant of such atypical usage; for example, developers of reasoning engines which operate over Web data and consider RDF collections as part of complex OWL class descriptions – and even though we did not find such usage in our dataset – should implement simple checks to ensure that the respective engine is tolerant to cyclic, non-terminating and branching collection descriptions.

2.3.2 Use of undefined classes and properties

Category: **incoherent**

Oftentimes on the Web of Data, properties and classes are used without any formal definition. For example, publishers might say that `ex:Alice ex:colleague ex:Bob` even though `ex:colleague` is not defined as a property. Again, although such practice is not prohibited, by using ad-hoc undefined classes and properties publishers make automatic integration of data less effective and forego the possibility of making inferences through reasoning.

From our crawl, 1.78M triples (14.3% of all triples) use undefined properties, appearing in 39.7k documents (72.4% of valid RDF/XML documents): Table 1 enumerates the top five.[10]

For example, from our crawl, the `livejournal.com` domain uses the properties `foaf:member_name` and `foaf:tagLine` in a total of almost 300k triples[11] – the FOAF vocabulary does not contain these properties and they are not defined elsewhere; such a practice of deliberately inventing undefined properties within a related namespace is common on the Web. Sometimes publishers make simple spelling mistakes: again, the property `foaf:image` is incorrectly used instead of `foaf:img` in the `livejournal.com` domain; to take another example, the term `qdoslf:neighbour` is commonly used – in 100k triples – instead of the property `qdoslf:neighbours` defined in the namespace.[12]

Undefined Property	Triples Used
`foaf:member_name`	148,251
`foaf:tagLine`	148,250
`foaf:image`	140,791
`cycann:label`	123,058
`qdoslf:neighbour`	100,339

Table 1: Count of the top five properties used without a definition

Undefined Class	Triples Used
`sioc:UserGroup`	21,395
`rss:item`	19,259
`linkedct:link`	17,356
`politico:Term`	14,490
`bibtex:inproceedings`	11,975

Table 2: Count of the top five classes used without a definition

Similarly, there were 1.01M triples (8.1%) mentioning undefined classes in 21.3k documents (38.8%); the top five instantiated such classes are enumerated in Table 2. Neither of the first three classes nor the last class are defined in the dereferenced documents; for example, all of the `sioc:UserGroup` instances come from the `apassant.net` domain[13]. To take another example, the class `politico:Term` is generically described in the dereferenced document, but is neither implicitly nor explicitly typed as a class.

Publisher Recommendations: Many such errors are indeliberate and due to spelling or syntactic mistakes resolvable through minor fixes to the respective ontologies or exporters. Where terms have been knowingly invented, we suggest that the term be recommended as an addition to the respective ontology – or defined in a separate namespace – to enable re-use.

Consumer Recommendations: Liberal consumer applications could, for example, use fuzzy string matching techniques – e.g., Levenstein distance measures – between undefined classes and properties encountered in the data, and classes and properties defined in the vocabularies. Generally however, consumer applications can usually overlook such mistakes and simply accept the consequence of incomplete reasoning for triples using such undefined terms.

2.3.3 Misplaced classes/properties

Category: **incoherent**

Sometimes, a URI defined as a class is used as a property (appears in the predicate position of a triple) or, conversely, a URI defined as a property is used as a class (appears in the object position of an `rdf:type` triple); although not prohibited, such usage is usually inadvertent and can ruin the machine-interpretation of the associated data.

[7] `http://scripts.mit.edu/~kennylu/myself.rdf`

[8] cf. `http://iswc2006.semanticweb.org/submissions/Harth2006dq_Harth_Andreas`

[9] `http://web.mit.edu/dsheets/www/foaf.rdf`

[10] It is important to note that herein, when we mention "undefined" classes or properties, we loosely refer to classes or properties "not defined in our crawl". In any case, our crawl would contain any property- or class-descriptions published according to best practices (i.e., using dereferencable terms).

[11] cf. `http://danbri.livejournal.com/data/foaf`

[12] cf. `http://foafbuilder.qdos.com/people/danbri.org/foaf.rdf`

[13] cf. `http://apassant.net/home/2007/12/flickrdf/data/people/36887937@N00` – indeed the authors herein are also prone to making simple errors in their publishing.

Class	# Misplaced
rdfs:range	8,012
foaf:Image	639
rdfs:Class	94
wot:PubKey	18
foaf:OnlineAccount	15

Table 3: Top five "classes" used in the predicate position of a triple

Property	# Misplaced
foaf:knows	4
foaf:name	4
foaf:sha1	2
swrc:author	1
foaf:based_near	1

Table 4: Top five properties found in the object position of an rdf:type triple

Table 3 shows the top five classes used as a property in our crawl. In fact, rdfs:range is a core RDFS property, but is defined in one document[14] as a class; hence the 8,012 occurrences are valid use of the property and the single declaration of rdfs:range as a class is at fault (this is also an instance of ontology hijacking, which we will discuss in Section 2.4). Most occurrences of the foaf:Image class used as a property stem from the sembase.at domain[15]; here the foaf:depiction property would be more suitable. Use of rdfs:Class as a property comes from the ajft.org and rdfweb.org domains[16] where rdfs:Class is seemingly mistaken as rdf:type. The class wot:PubKey is mistakenly used instead of wot:hasKey[17]. Misuse of foaf:OnlineAccount stems from one document[18] wherein the RDF/XML shortcut rdf:parseType="Resource" is used inappropriately, causing parsing of foaf:OnlineAccount elements as predicates.

After reasoning, more such errors were discovered, particularly in the affymetrix.com domain[19] which describes genes and mistakingly uses rdfs:subClassOf to assert subsumption relations between properties (amongst many other issues); this resulted in properties – which, combined, were used in 37,454 triples – being typed as classes.

Conversely, the usage of properties in the class position – viz. the object position of an rdf:type tripe – is much less common; Table 4 lists the results, with most errors stemming from one document[20].

Publisher Recommendations: Again, all such errors could easily be fixed by the publishers once they are made aware. Many of the above encountered errors were as a result of misuse of RDF syntactic terms, such as rdf:parseType="Resource", or more generally as syntactic mistakes in their documents: thus, publishers should not only ensure that their documents are syntactically valid, but also that they parse into the triples expected.

Consumer Recommendations: Applications which incorporate reasoning should consider foregoing standard inferences which rely on the position of a term in a triple to infer that that term is a class or property – for example, rule rdf1 in RDFS [10]. Aside from this, consumer applications will probably have to accept incomplete inferencing over such erroneous triples.

[14] http://www.w3.org/2000/10/swap/infoset/infoset-diagram.rdf

[15] cf. http://wiki.sembase.at/index.php/Special:ExportRDF/Dieter_Fensel

[16] cf. http://swordfish.rdfweb.org/discovery/2004/01/www2004/files/1101776794087.rdf

[17] cf. http://www.snell-pym.org.uk/alaric/alaric-foaf.rdf

[18] cf. http://tommorris.org/foaf

[19] cf. http://affymetrix.com/community/publications/affymetrix/tmsplice/all_genes.1.rdf

[20] http://www.marconeumann.org/foaf.rdf

D.type Prop.	# Non-literal	% Non-literal
swrc:journal	19,853	97.8%
swrc:series	14,963	97.3%
ical:location	4	2.6%
foaf:name	4	∼0%
foaf:msnChatID	3	0.4%

Table 5: Top five datatype-properties used with non-literal values

Obj. Prop.	# Literal	% Literal
affy:startsAt	6,234	100%
affy:stopsAt	6,234	100%
affy:cdsType	5,193	100%
affy:frame	4,882	100%
affy:commonToAll	4,814	100%

Table 6: Top five object-properties used with literal values

2.3.4 *Misuse of* owl:DatatypeProperty/owl:ObjectProperty

Category: **incoherent**

The built-in term owl:DatatypeProperty describes properties which relate some resource to a literal value, i.e., an "attribute" property (in terms of Object-Oriented Programming); similarly, the OWL term owl:ObjectProperty describes properties which relate one resource to another (i.e., a "relation" property). Oftentimes, attribute properties are used between two resources, and relation properties are used with literal values.

From our crawl, we found a total of 34.8k triples (0.3%) with datatype-properties given non-literal objects (in 1,194 [2.2%] documents across 9 domains). Table 5 lists the top five; the only significant errors stem from l3d.de[21] which exports RDF from the Digital Bibliography & Library Project (DBLP) – they define two datatype-properties in the swrc: namespace but only use the properties with non-literal objects.

Analogously, there were 41.7k triples (0.3%) with object-properties given literal values (in 4,438 [8%] documents from 91 domains). Table 6 lists the top five; many such occurrences come from the affymetrix.com domain which commonly uses five different object-properties with literal values (in a total of 27.4k triples from our crawl). However, there were many other such properties with significant misuse including miscellaneous properties from the opencyc.org domain (6,161), foaf:page (3,160), foaf:based_near (1,078), ical:organizer (456), amongst others; again, the errors were spread over 92 different domains. In fact, the property foaf:myersBriggs (in the popularly used FOAF specification itself) was until recently incorrectly defined as an owl:ObjectProperty with rdfs:range rdfs:Literal and had 35 literal values in our dataset.

Publisher Recommendations: Where datatype- or object-property constraints are erroneously specified – e.g., swrc:journal, swrc:series, foaf:myersBriggs – they can simply be reversed by the ontology maintainers. However, in many cases such constraints are purposefully defined to ensure consistent usage of the term; in this case, the onus is on publishers to thereby abide.

[21] cf. http://dblp.l3s.de/d2r/data/publications/conf/aswc/HoganHP08

Consumer Recommendations: Applications would typically use such constraints for form generation in the context of instance data creation. Liberal versions of such applications may decide to automatically reverse such constraints, where – in examples such as the `affy:` properties above – all usage is contrary to the specified constraint. Indeed, some weighting scheme may be adopted for examples – such as the `swrc:` properties – where *most* usage is contrary to the vocabulary constraint. Again, such approaches would admittedly need evaluation in the setting of the given application.

2.3.5 Members of deprecated classes/properties

Category: **incoherent**

Briefly, the OWL classes `owl:DeprecatedClass` and `owl:DeprecatedProperty` are used to indicate classes or properties that are no longer recommended for use: vocabulary publishers usually assert deprecation for classes or properties which have been considered to be obsoleted by more popular terms in local or remote vocabularies, or perhaps even where the original term is contrary to some naming scheme or considered outside of the scope of the vocabulary. In our dataset, we did not find any members of a deprecated class; however, we found 290 instances (in 115 documents) of four deprecated properties: `wordmap:subCategory` (260), `sioc:has_group` (15), `sioc:content_encoded` (10) and `sioc:description` (5).

Publisher Recommendations: Publishers of instance data should intermittently verify that no terms used have since been considered deprecated by the vocabulary maintainer, and should take appropriate action to use – possibly novel – recommended terms where possible.

Consumer Recommendations: Applications could consider specifying manual mappings from deprecated terms to compatible terms now recommended for use. Less liberal applications may consider omitting triples which use deprecated terms. Generally, however, usage of deprecated terms does not require special treatment.

2.3.6 Bogus `owl:InverseFunctionalProperty` values

Category: **incoherent/hijack**

Aside from URIs – which can be hard to agree upon – resources are also commonly identified by values for properties which uniquely identify a resource; such keys are pre-existing and easier to agree upon. These properties are termed "inverse-functional" and are identified in OWL with the term `owl:InverseFunctionalProperty`. If two resources share a common value for one of these properties, reasoning will view these resources as equivalent (referring to the same resource). For example, the FOAF ontology has defined a number of inverse-functional properties for identifying people; these include `foaf:homepage`, `foaf:mbox` (email), `foaf:mbox_sha1sum` (sha1 encoded email to prevent spamming), amongst others. Herein, FOAF holds the intuition that the values for such properties should be unique to an individual, and that the usage of such properties should reflect that (i.e., `foaf:mbox` should only be used for personal and unshared email-addresses).

However, FOAF exporters commonly do not respect the semantics of these inverse-functional properties and export 'void' values given partial user-input. The most widespread example is `08445a31a78661b5c746feff39a9db6e4e2cc5cf`, which is the encrypted SHA1 value of '`mailto:`' and is commonly assigned by FOAF exporters – as values for `foaf:mbox_sha1sum` – to users who don't specify an email in some input form.[22]

[22] In fact, at the time of writing, a Google search for this SHA1 string will result in nearly two million hits – seemingly almost all of which are FOAF RDF documents.

Now, all such users can be interpreted as equivalent – i.e., representing the same real-world person – according to the semantics of the `foaf:mbox_sha1sum` property. This problem is quite widespread: even in our diminutive crawl, 52 hosts contribute 1,169 different bogus values in 1,041 documents. For example, 194 errors come from the `bleeper.de` domain[23], 189 from `identi.ca`[24], 166 from `uni-karlsruhe.de`[25], 163 from `twit.tv`[26] and 92 from `tweet.ie`[27]; Table 7 details the top five void values for inverse-functional properties which we found in our dataset.

According to the standard reflexive, symmetric and transitive semantics of equality (represented in RDF by the equality relation `owl:sameAs`), if we take for example the 986 entries with the same null sha1 value, 986^2=972k `owl:sameAs` relations would be inferred. Further, assuming, for example, an average of eight triples mentioning each equivalent resource, 972k*8 = 7.8M statements would be inferred by substituting each equivalent identifier into each statement. In other words, such chains of equality cause a quadratic explosion of inferences; when one considers larger Web-crawls, the problem becomes quite critical.

Publisher Recommendations: For publishers, the issue is easily resolved by, for example, validating user input and checking the uniqueness and validity of inverse-functional values. Conversely, vocabulary maintainers should be careful to clearly state that a property is inverse-functional in the human-readable specification, and select labels for property URIs which give an indication of the inverse-functional nature of the property – for example, choose the label `ex:personalMbox` over `ex:mbox`.

Consumer Recommendations: A simple solution commonly used by reasoning agents is to simply blacklist void values. Although an exhaustive list of blacklist candidates may be difficult to derive, the above values would – in our experience – constitute most of the void values. Other heuristics may be employed to ensure correct equality reasoning – for example, use of a disambiguation step to quickly remove obviously incorrect equality inferences.

2.3.7 Malformed datatype literals

Category: **incoherent**

In RDF, a subset of well-defined XML datatypes are used to provide structure and semantics to literal (string) values. For example, string date values can be specified using the `xsd:date` datatype, which provides a lexical syntax for date strings and a mapping from date strings to date values interpretable by an application. From the content of the crawl, we found 3,666,840 literals of which 170,351 (4.6%) used a datatype. Of these, the top five most popular datatypes were `xsd:string` (53,879), `xsd:nonNegativeInteger` (38,501), `xsd:integer` (15,826), `xsd:dateTime` (15,824), and `xsd:unsignedLong` (12,318).

Unfortunately, incorrect use of datatypes is relatively common in the Web of Data. Firstly, datatype literals can be *malformed*: i.e., ill-typed literals which do not abide by the lexical syntax for their respective datatype. There were 4,650 malformed datatype literals (2.7% of all typed literals) in our crawl: Table 8 summarises the top five datatypes to be instantiated with malformed values.

The two most common errors for `xsd:dateTime` stem from

[23] cf. `http://bleeper.de/powerboy/foaf`
[24] cf. `http://identi.ca/whataboutbob/foaf`
[25] cf. `http://www.aifb.uni-karlsruhe.de/Personen/viewPersonFOAF/foaf_1876.rdf`
[26] cf. `http://army.twit.tv/takeit2/foaf`
[27] cf. `http://tweet.ie/seank/foaf`

Inverse-Functional Property	Void Value	Count
foaf:mbox_sha1sum	"08445a31a78661b5c746feff39a9db6e4e2cc5cf"	986
foaf:mbox_sha1sum	"da39a3ee5e6b4b0d3255bfef95601890afd80709"	167
foaf:homepage	<http://>	11
foaf:mbox_sha1sum	""	5
foaf:isPrimaryTopicOf	<http://>	2

Table 7: Count of the five most common void inverse-functional property values

Datatype	# Malformed	% Malformed
xsd:dateTime	4,042	26.4%
xsd:int	250	2.1%
xsd:nonNegativeInteger	232	0.6%
xsd:gYearMonth	67	100%
xsd:gYear	27	1.4%

Table 8: Top five datatypes having malformed values and percentage of all values which are malformed

Datatype Property	# Clashes	% Clashes
sl:creationDate	9,212	100%
scot:ownAFrequency	529	100%
owl:cardinality	464	65.2%
ical:description	262	21.8%
wn20schema:tagCount	204	100%

Table 9: Top five properties with datatype-clashes and percentage of all values which cause clashes

(i) the wasab.dk domain[28] whereby time-zones are missing the required ':' separator; and (ii) the soton.ac.uk domain[29] wherein the mandatory seconds-field is not specified. For xsd:int, almost all errors stem from the freebase.com domain whereby boolean values True and False are found[30]. For xsd:nonNegativeInteger, all stem from the deri.ie domain[31] where non-numeric strings are incorrectly used. Finally, for xsd:gYearMonth and xsd:gYear, all illegal usage comes from the dbpedia.org domain[32] where full xsd:dateTime literals are used instead.

Publisher Recommendations: Clearly, malformed literals are quite common. In all examples, the errors can be resolved by simple syntactic fixes to the publishing framework, or removing or changing the datatype on the literal; one can conclude – especially in the absence of a popular validator for datatype syntax – that publishers are simply not aware of such issues.

Consumer Recommendations: Although datatype-aware agents could incorporate heuristics to shoulder common mistakes – e.g., publishers commonly omit the mandatory seconds field from date-time literals – not all such mistakes can feasibly be accounted for. Again – and in cases where the issue next discussed does not apply – such literals can simply be interpreted as plain literals.

2.3.8 *Literals incompatible with datatype range*

Category: **incoherent/inconsistent**

Aside from explicitly typed literals, the range of properties may also be constrained to be a certain datatype, mandating respectively typed values for that property; e.g., one can say that the attribute property ex:bornOnDate has xsd:date values. A *datatype clash* can then occur if the property is given a value (i) that is malformed, or (ii) that is a member of an incompatible datatype. Table 9 provides counts of datatype clashes for the top five such properties.

The property sl:creationDate has the range xsd:date but all triples with sl:creationDate in the predicate position have plain-literal objects – all such usage originates from the semanlink.net tagging system[33]; please note that plain literals without language tags are considered as xsd:strings [10] and so are disjoint with xsd:date. The property scot:ownAFrequency is given range xsd:float but only ever used in the domain linkeddata.org[34] with xsd:integer objects; xsd:integer is a sub-type of xsd:decimal and is disjoint with xsd:float [4]. owl:cardinality is often used with plain-literal objects[35] contrary to the defined range xsd:nonNegativeInteger. The property ical:description – defined as having range xsd:string – is almost always instantiated with a plain-literal object (99.8%); however, only the 21.8% which use language tags constitute an inconsistency[36]. Finally, wn20schema:tagCount has range xsd:nonNegativeInteger but is only used with plain literals in the w3.org domain[37].

Publisher Recommendations: In all such cases, the root problem could be resolved if the vocabulary publisher removes the range on the property; in many cases such an approach may even be suitable: properties such as ical:description which are intended to have prose values should remove xsd:string constraints – optionally setting the range as the more inclusive rdf:PlainLiteral datatype to encourage literal values – and thus allow use of language tags. However, the majority of such datatype domain constraints are validly used to restrict possible values for the property and the onus is on data-publishers to thereby abide.

Consumer Recommendations: Again, liberal agents could consider changing the defined range of the property to reflect some notion of "common" usage. Also, although the usage of properties often does not reflect the defined datatype range, in our dataset we found that the literal strings were almost always within the lexical space of the range datatype and that they were just poorly typed. We only found two properties which were given objects malformed according to the range datatype (before, we were concerned with malformed literals given an *explicit* datatype): viz. exif:exposureTime with range xsd:decimal (given 49 plain literals with malformed decimal values in one document[38]) and cfp:deadline with range xsd:dateTime (given 3 plain literals with malformed date-time values in 3 documents[39]). Thus, in all but the latter cases, liberal software agents could ignore mismatches

[28] cf. http://www.wasab.dk/morten/2004/08/photos/1/index.rdf

[29] cf. http://rdf.ecs.soton.ac.uk/publication/10006

[30] cf. http://rdf.freebase.com/rdf/aviation/aircraft_ownership_count

[31] cf. http://www.deri.ie/fileadmin/scripts/foaf.php?id=320

[32] cf. http://dbpedia.org/data/1994_San_Marino_Grand_Prix.xml

[33] cf. http://www.semanlink.net/tag/rdf.rdf

[34] cf. http://community.linkeddata.org/dataspace/kidehen2/subscriptions/Kingsley_Feed_Collection/tag/rdf

[35] 425 of 464 such examples stem from http://bioinfo.icapture.ubc.ca/subversion/Cartik/Object-OWLDL2.owl

[36] cf. http://www.ivan-herman.net/professional/CV/W3CTalks.rdf

[37] cf. http://www.w3.org/2006/03/wn/wn20/instances/wordsense-act-verb-3.rdf

[38] http://kasei.us/pictures/2005/20050422-WCCS_Dinner/index.rdf

[39] cf. http://sw.deri.org/2005/08/conf/ssws2006.rdf – an example of errors admittedly generated by an author of this paper.

Disjoint Classes	# Instances
foaf:Agent ⊓ foaf:Document	502
foaf:Organization ⊓ foaf:Person	328
foaf:Document ⊓ foaf:Person	232
sioc:Container ⊓ sioc:Item	194
sioc:Item ⊓ sioc:User	35

Table 10: Top five instantiated pairs of disjoint classes

between an object's datatype and that specified by the property's range, parsing the literal string into the value space of the range datatype; however, caution is required when considering non-standard datatypes: consider if a property ex:temp has the datatype ex:celcius as range and is used with an ex:fahrenheit value – clearly the value should not be parsed as ex:celcius although in it's lexical space.

2.3.9 OWL inconsistencies

Category: **inconsistent**

The Web Ontology Language (OWL) includes features – such as defining disjoint classes, inequality between resources, etc. – which can additionally be used to check if some data agrees with the underlying ontology; i.e., that the data is *consistent*.

To begin with, we quickly mention inconsistency checks which we performed, but which did not detect anything in the crawl. Firstly, the class owl:Nothing is intended to represent the empty class, and, as such, should not contain any members; in our dataset, we found no directly asserted members of owl:Nothing. Also, an inconsistency can occur when owl:sameAs and owl:differentFrom overlap; again, however, we found no such examples in our crawl – in fact, we found no usage of owl:differentFrom in the predicate position of a triple. Similarly, although we found two instances of owl:AllDifferent/ owl:distinctMembers usage, none resulted in an inconsistency. Continuing, we also performed similar checks for instances of classes which were defined as complements of each other using owl:complementOf; however, again we found no owl:complementOf relations in our dataset. Briefly, we also performed simple checks for unsatisfiable concepts whereby, for example, one class is (possibly indirectly) both a subclass-of and disjoint-with another class: for each class found, we performed reasoning on an arbitrary membership of that class and checked whether any of the inferred memberships were of disjoint classes; however, we found no such concepts on the Web.

In fact, all inconsistencies we found in our crawl were related to memberships of disjoint classes. The OWL property owl:disjointWith is used to relate classes which cannot share members; disjoint classes are used in popular Web ontologies as an indicator of inconsistent information. For example, in FOAF the classes foaf:Person and foaf:Document are defined as being disjoint: something cannot be both. Resources can be asserted to be members of disjoint classes either directly by document owners, or inferred through reasoning. We only detected a small number of such direct assertions in our crawl – generally, a resource is asserted to be a member of one class in one document and a disjoint class in a remote document.[40]

However, after reasoning on our dataset, there were 1,329 occurrences of inconsistencies caused by disjoint classes; Table 10 enumerates the top five.

The most prominent cause of such problems stem from two incompatible FOAF exporters for LastFM data: the same resources are simultaneously defined as being of type foaf:Person in the opiumfield.com domain[41] and inferred to be members of foaf:Document in the dbtune.org domain[42]. Again, there are many other exporters and domains which contribute; for example, an exporter of Wikipedia data in the sioc-project.org domain[43] uses the same URI to identify users and the users' Wikipedia profile page.

Publisher Recommendations: Such problems with inconsistent data – especially those arising from multiple sources – may be quite difficult to solve. The obvious and lazy solution is to remove the disjointness constraints from the relevant ontologies; however, these constraints are intended to flag nonsensical or conflicting information and removing them clearly does not solve the root cause. Currently, the main observed cause for such inconsistencies is the use of incompatible naming schemes – using URIs to identify two completely different things – most often across different domains; agreement must be reached on what is an appropriate identifier for the contentious resource.

Consumer Recommendations: There are two standard approaches for handling inconsistencies in agents incorporating reasoning: **resolve** or **overlook**; the former approach – which requires 'defeating' the 'marginal view' – may not be so in tune with the open philosophy of the Web, where contradiction could be considered a 'healthy' symptom of differing opinions. Rule-based reasoning approaches have the luxury of optionally overlooking inconsistencies, where inconsistent data can simply be flagged (e.g., see OWL 2 RL rules in [7] with false consequences). However, tableaux algorithms are less resistent to inconsistencies and are tied by the principle of explosion: *ex contradictione quodlibet* (from contradiction follows anything); some works focus on *paraconsistent* reasoning – tableaux reasoning tolerant to inconsistency – although such approaches are expensive in practice (cf. [14]). In any case, in either rule- or tableaux-based approaches – and depending on the application scenario – inconsistent data may be pre-processed with those triples causing inconsistencies dropped according to some heuristic measures.

2.4 Non-authoritative contributions

2.4.1 Ontology-hijacking

Category: **incoherent/hijack**

In previous work, we encountered a behaviour which we termed "ontology hijacking" [12]: the redefinition by third parties of external classes/properties such that reasoning over data using those external terms is affected: herein – and loosely – we define the notion of an authoritative document for a term as the document resolved by dereferencing the term, and consider all other (non-authoritative) documents as third-party documents (please see [12] for a more exhaustive discussion). Web ontologies/vocabularies published according to best-practices are thereby the only document authoritative for the terms in their namespace.

In our dataset, we found that 5,211 document engaged in some form of ontology hijacking – most such occurrences were due to third party sources 'echoing' the authoritative definition of a class or property in their local ontology. However, we also encountered examples of third-parties redefining class/properties. As an example, we found one document which redefines the core property rdf:type – defining nine of its *properties* as being the domain of rdf:type – effectively

[40] http://apassant.net/blog/2009/05/17/inconsistencies-lod-cloud

[41] cf., http://rdf.opiumfield.com/lastfm/profile/danbri

[42] cf., http://dbtune.org/last-fm/danbri.rdf

[43] cf. http://ws.sioc-project.org/mediawiki/mediawiki.php?wiki=http://en.wikipedia.org/wiki/User:Andy_Dingley

leading to every entity described on the Web being inferred as a member of those nine properties.[44] Again, for example, we found 219 statements declaring `foaf:Image` – authoritatively defined as a class – to be a property; these were from the `sembase.at` domain (again see Footnote 15).

Publisher Recommendations: This particular issue focuses on how vocabulary publishers re-use existing vocabularies: we would thus particularly encourage vocabularies to extend external terms, and not redefine them. Such usage is more generally related to the principle of *modularity*, encouraging the modular design of Web vocabularies and avoiding the mess implied by the cross-definition of terms over the Web.

Consumer Recommendations: Clearly, on the Web, people should not be constrained in what they express and where they express it; however, to do useful reasoning, developers must take contextual information into account and provide some means of insulating ontologies from wayward external contributions. Again, in previous work we have described our system for performing reasoning over RDF Web data called SAOR [12], and found it essential to introduce our notion of authority when doing reasoning: in particular, we define our notion of an "authoritative rule application" which will not produce inferences from non-authoritative triples which redefine external terms. An orthogonal approach to the same problem is that of "quarantined reasoning" described in [5], which loosely constitutes "per-document" reasoning, and scopes inferences based on a closed notion of context derived from the implicit and explicit imports of each input document, thus excluding third-party contributions (please see [12] for a more in-depth comparison).

3. RELATED WORK

Earlier papers analysing problems in RDF Web data and the uptake of standards mainly focus on the categorisation and validation of documents with respect to the various OWL species. In [1], the authors performed validation – based on OWL-DL constraints – for a sample group of 201 OWL ontologies which were all found to be OWL Full for mainly trivial reasons; the authors then suggested means of patching the ontologies to be OWL-DL conformant. A similar but more extensive survey was conducted in [19] over 1,275 ontologies; the authors provided categorisation of the expressivity and species and discussion related to patching of the ontologies. At the moment, we do not offer species validation for RDFS/OWL and our scope is much broader with respect to validation.

In [15], the authors describe common user errors in modeling OWL-DL ontologies. In [17], the authors describe some error checking for OWL ontologies using integrity constraints involving the Unique Name Assumption (UNA) and also the Closed World Assumption (CWA). Similarly, in [18], various errors and constraints are introduced for error checking; the primary contribution is the introduction of five 'incongruencies' (e.g., an individual not satisfying a cardinality constraint according to UNA/CWA) with cases, causes and methods of detection. However, all of these papers have a decidedly more OWL-centric focus than our work and provide no analysis or discussion of Web data.

In [6], the authors provided an in-depth analysis of the landscape of RDF Web data in a crawl of 300M triples. Also they identified some statistics about classes and properties (SWTs) in RDF data; e.g., they found that 2.2% of classes and properties had no definition and that 0.08% of terms had both class and property meta-usage. However, again our focus is much more broad in characterising errors in RDF Web data.

[44] `http://www.eiao.net/rdf/1.0`

4. WHAT ABOUT ALICE?

We can now see that although our protagonist Alice is purely hypothetical, her adventures in Linked Data wonderland are disappointingly less so; in our analysis, we have shown the types of issues in RDF data on the Web that have made her journey so disconcerting. We have presented, provided statistics and examples for, and discussed a plethora of different types of errors, hopefully raising awareness of such issues amongst data publishers and developers of agents who wish to access and interpret such data. As typified by Alice, such issues can dramatically lower the quality of some applications, and consequently their end-user appeal; the errors do not come from the engine, but from the underlying data and thus, reasonable efforts to resolve data issues are as important as developing tolerant applications.

So, how can we help Alice?

We have already determined that many such issues are easily resolvable by the publisher and therefore concluded that publishers are unaware of the problems resident in their data. One solution would be to provide a system for validating RDF data being published to the Web: several systems exist but do not cover the broad range of issues discussed in this paper. From a syntactic point of view, the first validator available was the W3C RDF Validator[45], being able to check the syntax of any RDF/XML document (however, not datatype syntax). The DAML validator[46] provides checking of a large number of issues; however the validator is out of date (does not support OWL), and, at the time of writing, does not work. With regards to the protocol issues, the online Vapour validator[47] [3] aims at validating the compliance of published RDF data (either vocabularies or instances data) according to Linked Data principles [2]. The online Pellet [16] validator[48] enables species validation as well as other criteria we identified such as checking ontology consistency and finding unsatisfiable concepts.

There are also a number of command-line validators. The Validating RDF Parser (VRP)[49] operates on specified RDF Schema constraints, with some support for datatypes. The Eyeball[50] project provides command-line validation of RDF data for common problems including use of undefined properties and classes, poorly formed namespaces, problematic prefixes, literal syntax validation and other optional heuristics.

However, none of the above validators cover the plethora of issues we have encountered; thus, we have developed and now provide *RDF:Alerts*: `http://swse.deri.org/RDFAlerts/`. Given a URI, the system provides validation for *many* of the issues enumerated in this paper; Figure 2 shows a screenshot of feedback for an erroneous document. We further intend to extend the tool – to include *all* of the presented issues and suggestions from the community – and to improve usability; we may also consider extending such a tool to provide intermittent automatic reporting to publishers who opt in, depending on the perceived demand of such a service.

Still, other issues – particularly relating to inter-dataset incompatibility, naming, and inconsistent use of vocabulary terms – may be more difficult to resolve. Indeed, we have

[45] `http://www.w3.org/RDF/Validator/`
[46] `http://www.daml.org/validator/`
[47] `http://validator.linkeddata.org`
[48] `http://www.mindswap.org/2003/pellet/demo.shtml`
[49] `http://139.91.183.30:9090/RDF/`
[50] `http://jena.sourceforge.net/Eyeball/`

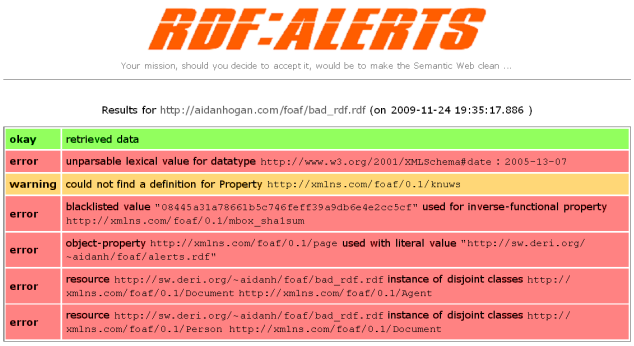

Figure 2: Screenshot of validation results from RDF:Alerts system.

also not properly discussed issues introduced by versioning, where, for example, a vocabulary maintainer makes changes to the definition of a term breaking backwards-compatibility with legacy usage of that term – indeed, we recognise that *casual versioning* may explain some of the discrepancies we have encountered in this paper, though systematic detection of such errors is difficult given our static snapshot dataset.

The resolution of such errors may sometimes require compromise between maintainers of ontologies and maintainers of exporters which populate the ontologies' terms, reflecting the current social and community driven nature of Web publishing. Reflecting such community driven efforts, consideration is being given to more open ontology editing and creation. In VoCamp events[51], people from different backgrounds and with different perspectives meet to work on modelling lightweight ontologies for immediate use. In order to allow ontologies to evolve according to user needs, initiatives such as semantic wikis for ontology management [13] and services such as OpenVocab[52] allow users to more freely interact with the ontology terms they wish to use and share. Although such approaches may again suffer from human error and disagreement – and have many open issues such as versioning and editing privileges – such community-driven efforts could lead to a more extensive vocabulary of terms for use on the Web.

We have also initiated a community driven effort which we call "The Pedantic Web Group"[53], which aims to engage with publishers and help them improve the quality of their data. Firstly, we have provided some pragmatic educational material for publishers, including a list of validation tools and of frequently observed problems in RDF publishing. Secondly, we have created a mailing list for actively contacting publishers about their mistakes and for various discussions on the quality of the Web of Data – subscription to which is open to the community. Indeed, such efforts may be the only means to resolve issues which require the co-ordination of multiple publishers. As such, we see the Pedantic Web Group as a go-to point for tackling publishing-related issues on the Web of Data, and as a community-driven means of promoting better quality publishing for the Web of Data.

To finally conclude, we would like to replace the present hypothetical Alice with a possible future Alice who is again browsing the Web of Data – however this time using an application which has been tempered for noisy data, where the documents have been validated, consistent identifiers used, and resources described using a rich vocabulary of community-endorsed terms. We hope that such an Alice might be amazed – this time for the right reasons.

[51] `http://vocamp.org/wiki/Main_Page`
[52] `http://open.vocab.org/`
[53] `http://pedantic-web.org/`

5. REFERENCES

[1] S. Bechhofer and R. Volz. Patching syntax in OWL ontologies. In *International Semantic Web Conference*, volume 3298 of *Lecture Notes in Computer Science*, pages 668–682. Springer, November 2004.

[2] T. Berners-Lee. Linked Data. Design issues for the World Wide Web, World Wide Web Consortium, 2006. `http://www.w3.org/DesignIssues/LinkedData.html`.

[3] D. Berrueta, S. Fernndez, and I. Frade. Cooking HTTP content negotiation with Vapour. In *Proceedings of 4th Workshop on Scripting for the Semantic Web (SFSW2008)*, June 2008.

[4] P. V. Biron and A. Malhotra. XML Schema part 2: Datatypes second edition. W3C Recommendation, Oct. 2004. `http://www.w3.org/TR/xmlschema-2/`.

[5] R. Delbru, A. Polleres, G. Tummarello, and S. Decker. Context dependent reasoning for semantic documents in sindice. In *Proceedings of the 4th International Workshop on Scalable Semantic Web Knowledge Base Systems (SSWS 2008)*, Karlsruhe, Germany, Oct. 2008.

[6] L. Ding and T. Finin. Characterizing the Semantic Web on the Web. In *Proceedings of the 5th International Semantic Web Conference*, November 2006.

[7] B. C. Grau, B. Motik, Z. Wu, A. Fokoue, and C. Lutz. OWL 2 Web Ontology Language: Profiles. W3C Working Draft, Apr. 2008. `http://www.w3.org/TR/owl2-profiles/`.

[8] A. Harth, J. Umbrich, and S. Decker. Multicrawler: A pipelined architecture for crawling and indexing semantic web data. In *5th International Semantic Web Conference*, pages 258–271, 2006.

[9] M. Hausenblas. Exploiting linked data to build applications. *IEEE Internet Computing*, 13(4):68–73, 2009.

[10] P. Hayes. RDF semantics. W3C Recommendation, Feb. 2004. `http://www.w3.org/TR/rdf-mt/`.

[11] T. Heath. How will we interact with the web of data? *IEEE Internet Computing*, 12(5):88–91, 2008.

[12] A. Hogan, A. Harth, and A. Polleres. Scalable Authoritative OWL Reasoning for the Web. *Int. J. Semantic Web Inf. Syst.*, 5(2), 2009.

[13] M. Krötzsch, S. Schaffert, and D. Vrandecic. Reasoning in semantic wikis. In *Reasoning Web*, pages 310–329, 2007.

[14] Y. Ma, P. Hitzler, and Z. Lin. Algorithms for Paraconsistent Reasoning with OWL. In *ESWC*, pages 399–413, 2007.

[15] A. L. Rector, N. Drummond, M. Horridge, J. Rogers, H. Knublauch, R. Stevens, H. Wang, and C. Wroe. Owl pizzas: Practical experience of teaching owl-dl: Common errors & common patterns. In *EKAW*, pages 63–81, 2004.

[16] E. Sirin, B. Parsia, B. C. Grau, A. Kalyanpur, and Y. Katz. Pellet: A practical OWL-DL reasoner. *Journal of Web Semantics*, 5(2):51–53, 2007.

[17] E. Sirin, M. Smith, and E. Wallace. Opening, closing worlds - on integrity constraints. In *OWLED*, 2008.

[18] J. Tao, L. Ding, and D. L. McGuinness. Instance data evaluation for semantic web-based knowledge management systems. In *HICSS*, pages 1–10, 2009.

[19] T. D. Wang, B. Parsia, and J. A. Hendler. A survey of the web ontology landscape. In *Proceedings of the 5th International Semantic Web Conference (ISWC 2006)*, pages 682–694, Athens, GA, USA, Nov. 2006.