Science Publishing and the Semantic Web: RSS, RDF and *Urchin* at Nature Publishing Group

Ben Lund
Nature Publishing Group
15th September 2004

Nature Publishing Group (NPG) is the publisher of *Nature*, the international weekly journal of science, and other research journals, review journals, references works, the online science news site news@nature.com, and, with the AFCS, the Signaling Gateway.

RSS and RDF at NPG

1. NPG's RSS feeds

   RSS is lightweight, XML-based, syndication format, version 1.0 of which is also an RDF vocabulary and serialization. NPG publishes RSS 1.0 Table of Contents documents for *Nature*, the *Nature Reviews* journals, all the *Nature*-branded research journals in the life sciences, and feeds for the news@nature.com science news site. As well as the standard RSS data fields of title, description and link, the RSS/RDF documents include extra bibliographic data using terms from the Dublin Core and PRISM (http://www.prismstandard.org/) RDF vocabularies.

   In addition, we publish three RSS/RDF feeds listing the most recent biological, physical and general scientific jobs advertised on Naturejobs (http://www.nature.com/naturejobs). These documents include job-specific metadata (employer, job title, location, posting date) using an RDF vocabulary developed in-house (http://www.nature.com/schema/2004/03/jobs.rdf). We are interested in working with other parties to expand and refine this vocabulary.

   A list of NPG's RSS feeds can be found at http://www.nature.com/rss.

2. Other Scientific RSS feeds

   Other scientific publishers are also beginning to issue RDF, carried in RSS 1.0 feeds. The Institute of Physics (http://syndication.iop.org/) , the International Union of Crystallography (http://journals.iucr.org/services/rss.html) and Ingenta (e.g. http://www.ingentaconnect.com/browsing/AllIssues?format=rss&journal=pubinfobike%3a%2f%2faiaa%2fjsr) all publish metadata-rich RSS/RDF. Below is an example of the data published in the *Nature* RSS feed.

```
<item rdf:about="http://dx.doi.org/10.1038/431113a">
  <title>
    Reviewers caution NASA over plans for nuclear-powered craft
  </title>
  <link>http://dx.doi.org/10.1038/431113a</link>
  <description>Tony Reichhardt,WASHINGTON</description>
  <dc:title>
    Reviewers caution NASA over plans for nuclear-powered craft
  </dc:title>
  <dc:creator>Tony Reichhardt</dc:creator>
  <dc:identifier>doi:10.1038/431113a</dc:identifier>
  <dc:source>Nature 431, 113 (2004)</dc:source>
  <dc:date>2004-09-09</dc:date>
  <prism:publicationName>Nature</prism:publicationName>
  <prism:publicationDate>2004-09-09</prism:publicationDate>
  <prism:volume>431</prism:volume>
  <prism:number>7005</prism:number>
  <prism:section>News</prism:section>
  <prism:startingPage>113</prism:startingPage>
</item>
```

RSS 1.0 is also being used to directly carry scientific data - CMLRSS, developed by Peter Murray-Rust and others, includes Chemical Markup Language (CML) data in feeds.

Urchin

1. Overview

Urchin is a web-based RSS aggregator and filter developed by NPG, initially funded by the UK Joint Information Systems Committee, and released as Open Source software (http://urchin.sf.net/). *Urchin* imports data from any version of RSS, and from non-RSS XML formats, and stores the data in a relational database. This data can be filtered to create new, content-specific RSS feeds and other arbitrary output formats. The content can be filtered using Boolean keyword searches, or by restrictions on specific data fields. For example, an *X-Prize* specific RSS feed could be creating by aggregating feeds from many different sources and filtered for occurrences of 'x-prize', 'scaled composites', etc. in item titles or descriptions. Or, an author-specific feed could be created by filtering on the dc:creator field. Below is an example of a cell biology news portal automatically created by aggregating and filtering RSS feeds.

2. RDF in Urchin

In addition to storing core RSS data in a relation database, Urchin stores all extra RDF data from incoming RSS 1.0 feeds in a rudimentary triple store. This has two advantages. First, all data that was associated with an item on import can be re-constructed when that item is included in a filtered output, and second, that data can be queried and filtered against. Urchin offers two forms of RDF filtering. There is a simple syntax that allows users to specify predicates and objects for items of interest. For example, a search of the form <dcterms:references>:=’doi:10.1038/nn0203-119’ would give a user a citation search RSS feed – in this case the feed would list all items that are known to cite the article with the Digital Object Identifier (http://www.doi.org/) ‘10.1038/nn0203-119’. Urchin also offers full RDF querying using the language provided by the RDF::Core::Query Perl module. This allows more complex filters to be created, and also allows direct querying of the database viewed as a triple store, rather than as a relational database.

3. Current development work

NPG is continuing to develop Urchin. We’re working on Bayesian filtering techniques to improve on the current Boolean keyword approach. We’re also using John Kleinberg’s burst detection algorithm (http://www.cs.cornell.edu/home/kleinber/bhs.pdf) to identify breaking stories by looking for jumps in the frequency of occurrence of words and phrases. We also plan to add the ability to import non-RSS RDF.

See the position paper ‘The Urchin-Kowari Project’ by David Wood, Taowei David Wang and Kendall Clark for further information about ongoing developments with *Urchin*.

Future plans

1. RSS expansion

   We plan to expand our use of RSS/RDF, both in terms of the number of feeds and the amount of metadata disseminated by each. Our immediate plans are to offer RSS 1.0 feeds for the non-*Nature* branded journals and for *NatureEvents*, our scientific events and conferences listing.

2. RDF Data services

   We are currently exploring the possibility of building richer data services on top of our current RDF-in-RSS offering. Using rdfs:seeAlso linking, we would publish a web of RDF documents that would expose data about our publications from the Journal to the individual article level. Our current thinking is that each individual article would have an RDF document associated with it, including bibliographic data, references lists, text analysis data and scientific metadata. Potentially, this would allow third parties to build services on top of our data – text mining and citation analysis are two obvious immediate applications.

3. *Copernicus*

   Copernicus is an internal NPG project to move towards capturing, publishing, and building services around rich data for each individual article we publish. As well as basic bibliographic and citation data, we could capture the following as structured data:
   - Scientific entities (genes, proteins, planets, fossils, etc) mentioned in the paper
   - 'Reason for citing' information
   - Conclusions
   - Raw results

   In addition, we could augment this data with information pooled from the web –gene and protein IDs from online databases, for example. Aggregating external data with our own would allow us to offer powerful search and query services.

Open Issues

1. RDF/XML serialization

   When building the richer data services outlined above, we intend to publish the data directly as RDF, and that will almost certainly mean offering RDF/XML documents. While NPG has no concerns about or difficulties with RDF's XML serialization, we know that many others do, and are concerned about possible barriers to the adoption of our services that those difficulties may create. There may be some scope here for work on guidelines for defining serialization restriction rules or RDF/XML profiles.

2. Cross-vocabulary mapping

   As we move towards pooling data from many different sources, we will encounter data

modeled using different RDF vocabularies, but expressing essentially the same underlying structure. We already encounter this in *Urchin* when mapping between the RSS 0.9, RSS 1.0 and Atom-in-RDF vocabularies. While the mapping can always be done programmatically, and on a case-by-case basis, beyond the simple assertion of equivalence between terms there is currently no clear, standard approach to mapping between these different vocabularies and structures.

3. Polling data sources and updating data stores

   There is currently no clear or general way of dealing with RDF data that is polled and stored from an individual source, and that changes over time. An application must decide whether the new data replaces, supplements, or conflicts with that currently held. The situation is somewhat analogous to the provenance problem, but is about comparing data from the same source at two different times, rather than from two different sources. At present, *Urchin* deals with this situation by wiping all previously stored triples before importing data, but this is clearly not a satisfactory situation. There does seem to be some scope for giving hints to applications – using OWL, for example, to describe property restrictions – but the details aren't yet clear.

4. Conclusions in RDF

   Some statements are very difficult to model in RDF. We encounter this difficulty when attempting to capture scientific conclusions as structured data. In particular, time- or context-dependent assertions are a problem – consider how the statements "When adenosine binds to A2A receptors, adenylyl cyclase is stimulated" or "The presence of caffeine leads to phosphorylation of Thr 75." could be rendered as RDF.

5. Community awareness

   It is increasingly important for the adoption of Semantic Web technologies that the disparate communities of technology vendors, scientists, librarians and publishers become more aware of the developments happening in each of these communities.

6. Tools and stores

   The current state of the art of tools for manipulating and storing RDF data is still relatively immature.