POWERING RSS AGGREGATORS WITH ONTOLOGIES
A Case for the RSSOwl Aggregator

Felipe M. Villoria, Oscar Díaz and Sergio F. Anzuola
The Onekin Group, University of the Basque Country, Pº Manuel de Lardizabal, 1 - 20018, San Sebastián, Spain

Keywords: RSS, Content Syndication, Ontology, Aggregation, Web Application.

Abstract: Content syndication through RSS is gaining wide acceptance, and it is envisaged that feed aggregators will be provided as a commodity in future browsers. As we consume more of our information by way of RSS feeds, search mechanisms other than simple keyword search will be required. To this end, advances in semantic tooling can effectively improve the current state of the art in feed aggregators. This work reports on the benefits of making a popular RSS aggregator, RSSOwl, ontology-aware. The paper uses three common functions, namely, semantic view, semantic navigation and semantic query, to illustrate how RSS aggregators can be "ontology powered". The outcome is that location, browsing and rendering of RSS feeds are customised to the conceptual model of the reader, making RSS aggregators a powerful companion to face the "RSSosphere". The system has been fully implemented, and successfully tested by distinct users.

1 INTRODUCTION

Most aggregators limit themselves to keeping feeds into folders, and little assistance is given to locate the desire news, except basic keyword search. As we consume more of our information by way of RSS feeds, the inability to store, index, and precisely search those feeds becomes more tiresome. This work strives to make aggregators ontology-aware. That is, feed providers continue to supply dull content, but the feed aggregator is now capable to annotate this content in accordance to a user-specific ontology.

As a prove of concept, this paper describes how a feed aggregator tool, *RSSOwl* (The RSSOwl Development Team, 2004), has been extended to permit feed consumers to provide their own ontologies. This accounts for enhancements in searching and tagging. Searching wise, the ontology permits a quicker and more focused location of the news of interest. As for tagging, news can be tagged with references to the ontology. Besides, when looking for a concept in the ontology, it is useful to locate not only the concept as such, but also those related concepts that are "semantically" related. Tagging is then extended not only to the sought concept but also to its neighbour concepts.

Related works include the *Artequakt* project (Alani et al., 2003) and the *SCORE* (Semantic Content Organization and Retrieval Engine) project (Sheth et al., 2002). The *Artequakt* project links a knowledge extraction tool with an ontology to achieve continuous knowledge support and guide information extraction. The extraction tool searches online documents and extracts knowledge that matches the given classification structure. As for *SCORE*, it provides a tool that focuses on automatic classification and metadata extraction based on semantic techniques.

The rest of this article motivates the need for ontologies (Section 2) and the advantages brought by the use of ontologies in feed aggregators using *RSSOwl* as a testbed (Section 3).

2 WHY ARE ONTOLOGIES NEEDED?

Traditional feed aggregators timely recover items. Search mechanisms can be available to locate items that contain a certain word. This approach poses at least two main drawbacks: lack of context and lack of abstraction.

Lack of context. Searches based on words rather than the concepts which those words denote, lead to the retrieve of inputs which are hardly related with the concept you are looking for. If you are interested

M. Villoria F., Díaz O. and F. Anzuola S. (2006).
POWERING RSS AGGREGATORS WITH ONTOLOGIES - A Case for the RSSOwl Aggregator.
In *Proceedings of the Eighth International Conference on Enterprise Information Systems - SAIC*, pages 197-200
DOI: 10.5220/0002458001970200

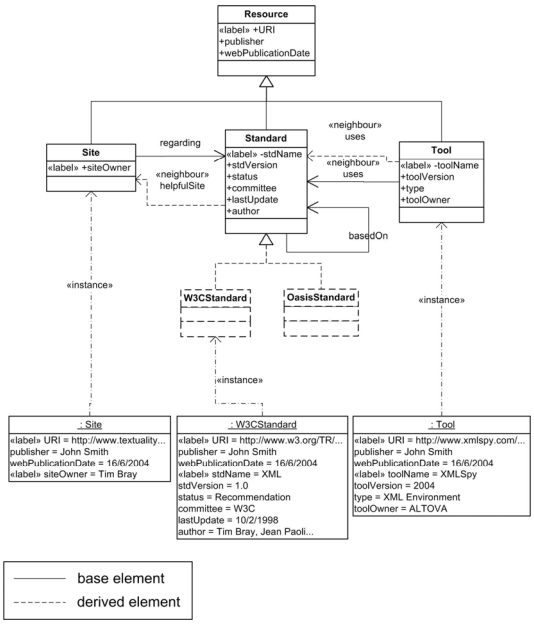

Figure 1: The running ontology on resources.

in owls, a word-based search will locate those items with the string "*owl*" inside. This could retrieve items on owls (the user's intent) but also on the *Ontology Web Language (OWL)*. The ontology fixes the concepts of a certain subject area which can be realised by distinct words, and provides the context to map a word with the intended concept.

Lack of abstraction. Ontologies enhance the level of abstraction at which queries can be posed. By introducing taxonomies and relationships, more abstracted concepts can be used that those explicitly appearing on the documents. For example, without the aid of a controlled vocabulary one may wonder whether to use the term "car", "automobile", or "vehicle" in performing a given search on the Internet. Backed by an ontology, however, the searcher may be advised that "automobile" should be used instead of "car". The degree to which terminology is semantically precise will have a direct impact on the degree to which relevant information can be found.

This work uses OWL for the description of the ontology. Figure 1 depicts the ontology that is used throughout this paper. It conceptualises the notion of resource for a software company. A resource can be either the tools being used by the organization, the standards which are followed by this company, or web sites that provide relevant material. Besides the *base classes*, *derived classes* can be defined by restricting the property values of a *base class*. For instance, the *W3CStandard* is a derived class of the *Standard* class which sets the property *committee* to *W3C*.

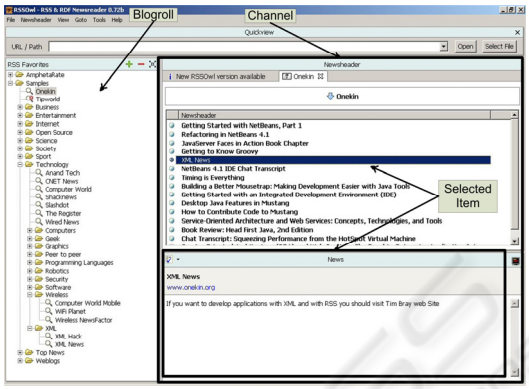

Figure 2: Starting scenario. The "Item" canvas is just text.

3 LEVERAGING RSSOWL

News are syndicated through feeds. A *feed* comprises a *channel*, which has a *title*, *link*, *description* and (optional) *language*, followed by a series of *items* (also known as news), each of which have a *title*, *link* and *description*. The content itself appears in the <*description*> tag, either as an HTML-escaped string or as a CDATA element. In other words, the description can not be annotated.

RSSOwl is a popular feed aggregator, free of charge, open-source and cross-platform (The RSSOwl Development Team, 2004). This work uses *RSSOwl* version 0.72b. Figure 2 shows the kickoff screenshot of *RSSOwl*. The upper-left hand side of the screen shows a folder hierarchy where folders corresponds to *blogrolls* (i.e. collections of Web log feeds that indicate the site you are subscribed to). Selecting a folder leads to display the items' titles from this channel on the right-hand side of the screen (rendered as anchors). From then on, the content of a single item (i.e. the <*description*> tag) can be obtained by selected any of these anchors. Basically, this situation can be characterised as channel grouping, keyword search, and folder-based navigation.

Ontology-aware aggregators can now leverage this functionality that would be difficult or impossible otherwise. Next paragraphs show how this has been achieved in RSSOwl that can now be configured with a OWL file.

Semantic query. Rather than a basic keyword search, a "*semantic search*" assists the user to map words in the syndicated news to concepts in the ontology. To obtain these benefits, *RSSOwl* is

empowered with user-provided ontologies. The addition of the ontologies is reflected as a new canvas on the screen. Specifically, this canvas is rendered at the lower-left hand side (see figure 3). The ontology is depicted as a specialization tree.

This canvas is used as a rudimentary query interface: the user clicks on the concept sought (e.g. the RSS instance) and the canvas on the right shows all news where this concept appears, *regardless of their container blogrolls*.

This output requires a mapping between the strings found on the news and the concepts of the ontology. However, instances of the ontology are identified through URIs, and these URIs will never be found in the news. What is needed is a human-readable version of the instance name.

To attain this aim, the notion of label is used. RDFS provides a *RDFS:LABEL* as a human-readable version of resource name. Analogously, one wants to assign an instance with a human-readable name even if it instantiates a class from a given ontology that does not use the property *RDFS:LABEL per se*. For instance, one might want to state that the property *siteOwner* of the *Site* class will serve to ascertain the appearance of a site in the news so that the appearance of the string "*Tim Bray*" in a feed will be taken as a reference to the resource whose owner is Tim Bray. That is, the *siteOwner* plays the role of the label for sites.

LABEL is then a kind of property. Data-based properties (e.g. *siteOwner*) can then be of the special kind label. Figure 1 depicts this through a tagged value which is associated with the property. Notice how a different label can be stated for each class: *stdName* is the label for *Standards* whereas *siteOwner* is the label for *Site*. It is also worth mentioning that a class can have more than one label, either defined among its own properties or inherited from its superclass. In our running example, all resources (should it be sites, standards or tools) have as one of their labels the property *URL* which is inherited from the resource class. Hence, the appearance of either "*Tim Bray*" (the owner) or "*www.textuality.com*" (the URL) will lead to a connection with the very same resource in the ontology.

For the domain at hand, basic word search seems to be sufficient as both polysemy and synonymy are rare, that is, "*XML*" or "*Tim Bray*" univocally identified the standard concept, and the *siteOwner* of the *www.textuality.com* site. This is due to the smaller amount of documents to be tagged and their more focused nature. Unlike other works where tagging is extended to the whole web (Sheth et al., 2002), this work focuses on tagging only those feed's items which have been previously selected by the user. This implies that the context is much more focused and hence, Tim Bray other than the owner of the XML site would hardly appear in the feed's item.

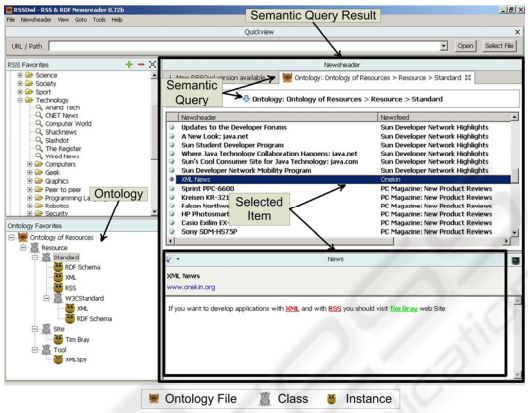

Figure 3: Semantic query enhancement.

Being ontology-powered, *RSSOwl* queries can now be stated at a higher level of abstraction. The user clicks on any of the concepts that appear on the lower-left hand side canvas, and the system displays all news related with the selected concept. For instance, it suffices a single click on *Standards* for *RSSOwl* to retrieve all news which contain the labels associated with the concepts *Standards*, *W3CStandards*, *RDF Schema*, *RSS* and *XML*, which would require several clicks otherwise (see figure 3).

Semantic view. Semantic metadata can also help to provide a view more akin to the aggregator's perspective rather than feeds being clustered according to their providers. Readers could thus arrange feeds according to their personal categorization schema, or being rendered along indexing schema that fit the reader's mental model.

For instance, if writing a report on XML, you could be interested in highlighting the *titles* of those items addressing XML issues, not to be overlooked in the bulk of everyday news. In general, highlight filters can be defined based on the presence/absence of concept's labels.

Another useful outcome is to annotate the *content* rather than the title of the news. Figure 4 shows this situation. An item has been selected. Its content is annotated, i.e. the string "*XML*" is recognised as the label of the XML concept. Rendering wise, this leads this string to be highlighted, and turned into an anchor to the URL of the associated ontological term (e.g. *http://www.w3.org/TR/1998/REC-xml-19980210*). In this way, news items are enhanced with links to the corresponding URLs, and in so doing, the user

can directly access the resource site for further information.

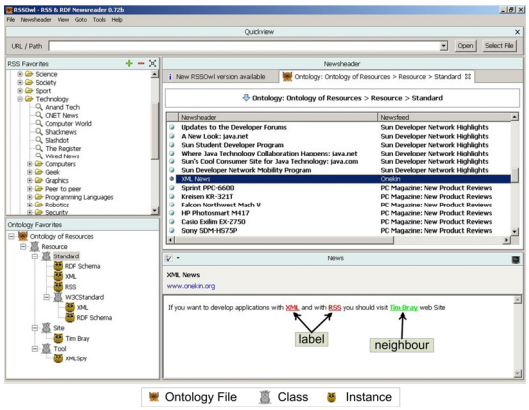

Figure 4: Semantic view enhancement.

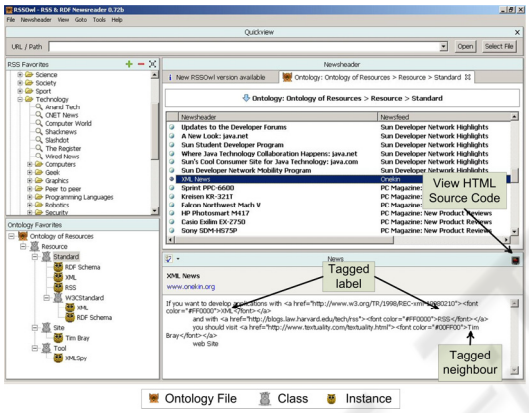

Figure 5: Semantic navigation enhancement: the "Item" canvas becomes a hypertext.

Semantic navigation. Semantic metadata allows for semantic browsing, i.e. you can move along semantically related feeds. For instance, once positioned in a feed which is annotated with a given *XML* ontological term, you can be interested in moving to those feeds that are "semantically" related. Hence, searching for the XML concept will locate not only the "XML" string, but also semantically related concepts such as the site of "*Tim Bray*" (see figure 5).

This notion of "semantic proximity" is realised through the *neighbour* association. An association plays the role of a neighbour for a given class, if the content of this property points to resources that are closer to the resource at hand. Consider again the sample ontology. The *helpfulSite* property plays the role of *neighbour* for the *standard* class. This implies that if looking for the XML standard, the system will tag the "*XML*" string as well as the strings that correspond to labels of sites that are helpful for *XML* (i.e. the content of the *helpfulSite* property).

This situation is shown in figure 5. The bottom-right hand side of the screen continues to display the content of the selected news. Now, however, the raw content provided by the news is "markuped" with the neighbour concept. To better assess the implementation, figure 5 shows the source code:

```
If you want to develop applications
with <a href="www.w3.org/TR/1998/REC-
xml-19980210"> <font color="#FF0000">
XML </font> </a> and with <a
href="http://blogs.law.harvard.edu/tech
/rss"> <font color="#FF0000"> RSS
</font> </a> you should visit <a
href="http://www.textuality.com/textual
ity.html"> <font color="#00FF00"> Tim
Bray </font></a> web Site
```

An anchor markup wrappers "*Tim Bray*" as an indication that his website is a *helpfulSite* for the *XML* concept.

In this way, the plain text of the *RSS* <*description*> tag is now turned into hypertext. The user is no longer limited to browse items based on their repository folders. Rather, a folder-cutting navigation is facilitated where the user moves from one item to those ontologically related items.

4 CONCLUSIONS

This work describes how semantic tooling can be successfully applied to RSS aggregators. The *RSSOwl* aggregator is used as a testbed. The tool is currently at work in a research lab. Users particularly appreciate the ability to search at a higher-level of abstraction, and the tagging of neighbours.

REFERENCES

Alani, H., Kim, S., Millard, D. E., Weal, M. J., Hall, W., Lewis, P. H., and Shadbolt, N. R. (2003). Automatic Ontology-Based Knowledge Extraction from Web Documents. *IEEE Intelligent Systems, (January-February):2-9*

Sheth, A., Bertram, C., Avant, D., Hammond, B., Kochut, K., and Warke, Y. (2002). Semantic Content Management for Enterprise and the Web. *IEEE Internet Computing, (July-August). to appear.*

The RSSOwl Development Team (2004). Welcome to RSSOwl - Free RSS & RDF Newsreader. *http://www.rssowl.org.*