semiBlog – Semantic Publishing of Desktop Data

Knud Möller, John Breslin and Stefan Decker

DERI, National University of Ireland, Galway, Republic of Ireland.
(knud.moeller, john.breslin, stefan.decker)@deri.org

Introduction

A number of approaches have been suggested to solve the chicken-and-egg problem that the Semantic Web faces; one of the most prominent among them being the automatic generation of formal data from text and other sources through the use of various Information Extraction (IE) techniques. In this paper, however, we will not take that approach, but instead follow the suggestion to build applications that would produce formal, Semantic Web (SW)-conformant data as a by-product of processes that users might already be performing on a regular basis [6].

A typical user will often have a vast amount of data available on their desktop[1], such as entries in an electronic address book, bibliographic metadata in formats like BibTeX, or MP3 metadata. This data might either have been entered manually by the users themselves, or collected externally (e.g. through sources like Bibster [5] or Gracenote's CDDB[2]). Such data is already formal data, and can in principle be transformed into a SW format like the Resource Description Framework (RDF). However, while this transformation is theoretically a simple enough process, in the absence of easy-to-use tools it is time consuming and tiresome for an ordinary user to perform. This leads to the situation that there is usually little to no motivation to actually produce and publish SW data. The data remains locked within the user's computer.

At the same time, the recent phenomenon of weblogging (or "blogging") [14] has made it possible for ordinary users to publish on the Web, and thus become content producers instead of content consumers. A number of blogging platforms such as Blogger[3] or Movable Type[4] allow users to pub-

[1] We use the term *desktop* as a metaphor for the entire working environment within a computer.
[2] CDDB: www.gracenote.com/music/
[3] Blogger: www.blogger.com
[4] Movable Type: www.sixapart.com/movabletype/

Advances in Information Systems Development: Bridging the Gap between Academia and Industry. Edited by A.G. Nilsson et al., Springer, 2005

lish almost any kind of data (mostly text and pictures). Blog authors manage their own content in their own blog, structure it through time in the form of discrete blog entries and are able to categorise these entries. It is also possible to comment on other people's entries or refer to them through links (trackbacks[5], pingbacks[6]). Most blogs provide a so-called newsfeed, which acts as a syndicated table of contents. These feeds are usually published alongside the blog at a separate URL and can contain various kinds of metadata for each entry (e.g. a title, short description, date, author, link, etc.). Feeds are used by blog or news readers and aggregators to allow subscription to and aggregation of different blogs. Various kinds of dialects for newsfeeds are currently in use, the most prominent being the RSS family of feed languages. Of these, versions 0.9x and 2.0 are implemented in XML, while the 1.0 strand (RDF Site Summary (RSS 1.0) [1]) uses RDF[7].

Semantic Blogging

We believe that it will be possible to utilise the blogging phenomenon to unlock the data on a user's desktop and make it easy to publish it on the SW. The result would be what we call "semantic blogging" – blogs enriched with semantic, machine-understandable metadata. Whenever a user publishes an entry that relates to or discusses some entity that has a formal representation on his desktop, they could add an RDF version of this representation to the blog entry. For example, an entry mentioning one or more persons might be enriched with RDF versions of the respective address book entries, and entries about papers or other documents would then contain an RDF version of the corresponding BibTeX metadata. In this way, semantic metadata can be made available with little or no overhead. We can also assume that the user has a strong motivation for publishing a blog to begin with, so that as a result of these factors, the problem of motivation for creating SW data would disappear.

Semantic blogging has two main advantages: *(i)* the addition of semantic metadata would make content published in a user's blog easier to find and browse through, and *(ii)* this would help to solve the chicken-and-egg problem of the SW.

Building a semantic blog in the way described above would require software that integrates tightly with the user's desktop environment. In this pa-

[5] Trackbacks: www.movabletype.org/trackback/

[6] Pingbacks: www.hixie.ch/specs/pingback/pingback/

[7] The situation in the RSS world is a little complicated. Please refer to [10] for a more in-depth explanation of the various versions and strands.

per we describe the *semiBlog* system. semiBlog is similar to other software like MarsEdit[8] in that it allows a user to write blog entries, structure them through time, and add pictures to them – in short, to produce their own blog. However, semiBlog goes beyond the capabilities of existing software in that it allows the user to easily tap the data that exists on their computer, transform it into SW-conformant data and add it to the blog, thus producing a semantic blog.

Outline of the Paper

The remainder of this paper is structured as follows. We will begin by presenting some other recent approaches to semantic blogging. After this, we will discuss a number of possibilities as to how and where semantic metadata can actually be represented in a blog. This discussion is followed by an overview of the semiBlog prototype itself. We will describe a short example session of the editor, present in detail the architecture of semiBlog, and close with some thoughts concerning future work.

Other Approaches to Semantic Blogging

The concept of Semantic Blogging is not our invention; a number of recent papers have investigated the topic from different angles. [7] discusses a semantic blogging prototype built on top of the SW browser Haystack [12]. They interpret blog entries mainly as annotations of other blog entries and web resources in general, and devise a platform to realise this in terms of the SW. We extend the scope of this interpretation and include the possibility of annotating resources that originally do not exist on the Web, but only on a user's desktop. The paper also underlines the inherent semantic structure of blogs and their entries as such, and presents a way of formalising these semantics in a SW fashion. This point is also made (on a more general level, encompassing various kinds of web based information exchange such as e-mail, bulletin boards, etc.) by [2] and [11]. [4] puts a strong emphasis on the use of semantic technologies to enhance the possibilities of blog consumption, by allowing viewing, navigation and querying with respect to semantics. The paper describes a prototype for both creation and browsing of semantic blogs, which was developed as part of the SWAD-E project[9]. While the prototype only deals with bibliographic

[8] MarsEdit: http://ranchero.com/marsedit/
[9] SWAD-E: www.w3.org/2001/sw/Europe/

metadata as annotations to blog entries, the authors point out that the same technologies can be used for any kind of metadata. [8] describes a platform called Semblog, which uses the Friend of a Friend (FOAF) ontology [3] as an integral part. FOAF descriptions of blog authors are linked to their blogs. In this way, the blog as a whole is annotated with metadata about its author. On a more fine-grained level individual blog entries are classified by linking them to personalised ontology. To implement their platform, the authors provide both a Perl CGI-based tool called RNA and a standalone Windows-based tool called Glucose.

Where to Put the Semantics

When trying to answer the general question of how to actually add semantic metadata to blogs and thus turn them into semantic blogs, two problems have to be addressed: *(i)* the choice of format to express the semantics and *(ii)* the physical location for placing the metadata. From a SW perspective, the answer to the first problem is straightforward: since the fundamental data model and language for the SW is RDF, semantic metadata for blogs should use this format. However, we also generally believe that the graph structure of RDF makes it an excellent choice for expressing semantic relations and metadata, as opposed to the tree structure of simpler formats like XML. As to the second problem, there are two basic options, defined by the two publication channels which blogs usually use: the HTML rendered version and the syndicated feed. The HTML rendering is the main channel, which can be viewed by readers in a web browser. The newsfeed usually has the function of augmenting the HTML, and mostly acts as a summary or table of contents. As we have already mentioned, these feeds contain metadata about the blog and its entries and are implemented in either XML or RDF.

The first option for placing the metadata would then be to bind it to the HTML pages. Several approaches for achieving this have been discussed; a good overview is given by Palmer [9]. However, Palmer also mentions that many approaches seem to suffer from inherent disadvantages: embedding the RDF might either lead to invalid HTML or require long and complex DTDs. Also, the problem of conflicting fragment identifiers arises. Another approach for binding to HTML is to link to external RDF resources, by using the `<link>` tag (in the head of the document) or the `<a>` tag (in the body of the document). However, the first possibility will restrict the author to only have metadata relating to the whole HTML document, while the second might lead to confusing hyperlinks in the

browser rendering of the document. A number of other approaches are mentioned, but they all require significant implementation efforts in the form of specialised parsers, etc.

```
<rdf:RDF
  <!-- ... metadata about the blog in general, etc. -->>
  <rss:item rdf:about=
    "http://www.example.org/blog#YARSMeeting">
    <rss:title>YARS and Space Travel</rss:title>
    <rss:link>http://www.example.org/blog#YARSMeeting
    </rss:link>
    <rss:description>
      Today I had a meeting with Andreas. We went over his
      paper and talked about possibilities of using his
      YARS RDF store for a manned mission to Alpha Centauri
    </rss:description>
    <dc:date>2005-04-06</dc:date>
    <dc:subject>
      <foaf:Person rdf:ID="andreas">
        <foaf:homepage
          rdf:resource="http://sw.deri.org/~aharth/"/>
        <foaf:surname>Harth</foaf:surname>
        <foaf:firstName>Andreas</foaf:firstName>
        <!-- ... more properties ... -->
        <rdf:value>Andreas Harth</rdf:value>
      </foaf:Person>
    </dc:subject>
    <dc:subject>
      <bibtex:InProceedings>
        <bibtex:title>Yet Another RDF Store: Perfect Index
          Structures for Storing Semantic Web Data With
          Contexts
        </bibtex:title>
        <bibtex:author rdf:resource="#andreas"/>
        <!-- ... more properties ... -->
        <rdf:value>YARS Paper</rdf:value>
      </bibtex:InProceedings>
    </dc:subject>
  </rss:item>
  <!-- ... more entries ... -->
</rdf:RDF>
```

Fig. 1. Example of an RSS 1.0 feed, enriched with additional metadata

Another option – and in fact the option chosen by all approaches towards semantic blogging listed in the section on other approaches – is to place the semantics into the news feed. Using this option, the most natural choice of feed dialect is RSS 1.0. As one of the various RSS dialects, its most obvious purpose is to act as a simple table of contents for the blog. However, since it is based on RDF, it can easily be extended to accommodate arbitrary additional RDF metadata. As noted by [7] one problem of embedding metadata in newsfeeds is that they often only contain the *n*

most recent entries. The authors of this paper also note that the solution to this problem can be archives of newsfeeds. Despite possible problems, we chose to adopt this approach, mainly because it does not have any of the inherent problems of binding to HTML and will be compatible with ordinary RDF crawlers without any modification.

Figure 1 shows an example of an RSS 1.0 feed with additional metadata, using the `RDF/XML` syntax[10]. The `<rss:item>` tag indicates an instance of the `item` class and represents an individual blog entry; all enclosed elements represent properties of this entry. Properties in the `rss` namespace are defined in the core RSS 1.0 specification, while the `dc` namespace is reserved for the Dublin Core[11] extension module[12]. The `foaf` and `bibtex` namespaces are not part of the specification at all and represent examples of arbitrary additional metadata added to the RSS feed.

Due to the nature of RDF there are various possibilities to associate arbitrary metadata with a blog entry in RSS 1.0 , e.g. [4] defines a `contains` property, and [8] adopt the `foaf:topic` property for the same purpose. However, we chose to use the `dc:subject` property already included in the Dublin Core module. While the only valid values currently defined for this property are flat string literals, the specification also suggests the possibility of using richer semantics, by allowing complex resources instead of strings. We follow this suggestion, as illustrated in Fig. 1. In this example, the user has written a blog entry about a meeting he had with his colleague Andreas, concerning a paper written by him. Since the entry is therefore relating to Andreas and his paper (amongst other things), complex metadata about these two entities has been added to the blog's RSS feed. More specifically, it has been added as the entry's `dc:subject`. A "dumbed down" representation of the resources is also provided, using the `rdf:value` property.

semiBlog Application

The main focus in the creation of semiBlog was to make the authoring of a semantic blog as easy as possible. In particular, we wanted to allow the user to easily add semantic metadata to the blog about entities that already exist on their desktop. This led to the early design decision to make semiBlog a desktop-based application, rather than web-based. Access to

[10] RDF/XML: www.w3.org/TR/rdf-syntax-grammar/
[11] Dublin Core: http://dublincore.org
[12] For details on RSS 1.0 and modules, see http://web.resource.org/rss/1.0/modules/

other desktop applications and their data (e.g. through their public APIs), control of the clipboard, and techniques like drag-and-drop are difficult or impossible to implement in a web-based environment. Another design decision was to make the first prototype a native, platform-dependent application – this was chosen for similar reasons (access to application APIs, etc.), but also because it allowed for a much easier and quicker development process. For these reasons, semiBlog is currently only available for the Mac OS X operating system. Future versions of semiBlog could be more platform independent, particularly the components comprising the *semiBlog Core* block in Fig. 3. However, we believe that for an application that wants to interface directly to existing desktop data, a certain degree of platform dependence is always necessary.

Example Scenario

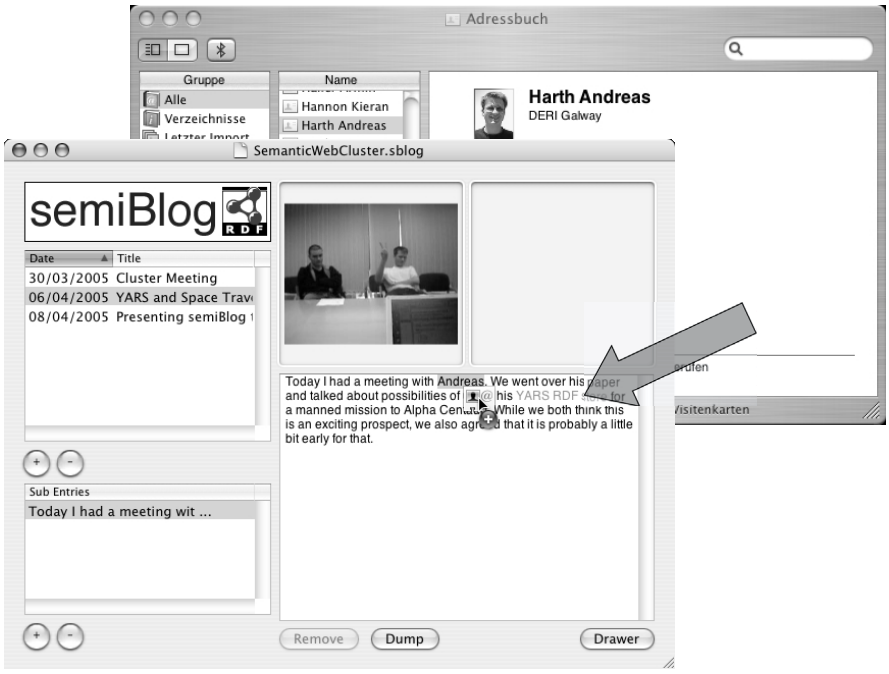

Fig. 2. Screenshot of the semiBlog editor

A screenshot of an example session in semiBlog is shown in Fig. 2. The user has just had a meeting with his colleague Andreas, where they went over an academic paper discussing one of Andreas' projects, named

"YARS". He creates a new entry for this in his blog, adds some text and a picture. Then he decides to annotate the entry with some semantic metadata. To do so, he simply selects the name of his colleague, drags the corresponding card from his address book and drops it onto the text field.

The entry (or rather a part of it) is now linked to a piece of data on the user's desktop. In a similar fashion he drags and drops the URL of the YARS project page from his web browser, as well as the BibTeX entry for the paper from his bibliography tool. Once a piece of text has been annotated, it is highlighted in the editor, so that each type of data is represented by a different colour.

Architecture and Flow of Data

Figure 3 gives an abstract overview of the architecture and flow of data in semiBlog. The left-hand side of the figure shows some examples of desktop data that a user might have. We show address book entries, bibliographical metadata and web pages, but any other kind of data is conceivable as well. semiBlog's architecture allows custom wrapper plug-ins for each data source, which take the object information from the various desktop applications and transform it into RDF metadata form. Together with the textual entry provided by the user, this metadata is then combined into an intermediate XML representation. To generate the actual blog, the XML is transformed into the various publication channels (currently a non-semantic HTML rendering and the semantic RSS 1.0 feed, which links to the HTML). The blog is now ready for publication on the web.

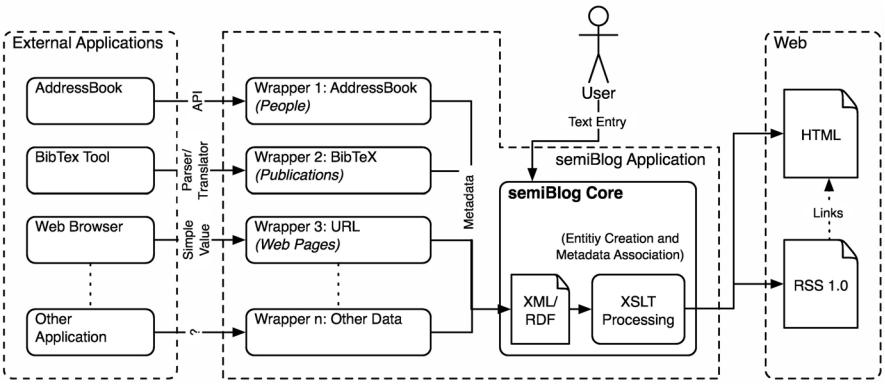

Fig. 3. semiBlog architecture and flow of data

Wrappers

For each individual wrapper, access to the data can be handled in a different way: if the data is tied to a specific application (e.g. the AddressBook application for Mac OS X), then access via an application API might be possible. Other, more generic data (e.g. BibTeX) will be handled by appropriate parsers/translators. Wrappers are implemented as plug-ins to the platform, which makes it easy to add new functionality and cover more data sources as the need arises. Plug-ins are basically implementations of an abstract `AnnotationType` class, which defines the basic API each plug-in has to provide. Parts of this API are getter and setter methods for the supported data types (objects in a drag-and-drop operation provide their data in various types or data flavors); colours for highlighting in the editor; a method `annotationFromData` to generate RDF; and a method `mainAnchorFromData` to generate a hyperlink reference for the HTML rendering of the blog.

Intermediate XML

semiBlog uses XML as an intermediate data format. Each entry is represented by an `entry` element, the textual content is contained in a `text` element. This element allows mixed content of text and `annotation` elements, which provides the possibility to add metadata to individual substrings of an entry. We chose this inline annotation technique over external annotation by reference, because it makes the XSLT transformations in the next step easier to accomplish. Each annotation contains a `mainAnchor` to be used in the HTML rendering of the blog (or in any other non-semantic representation), the RDF `metadata` and the actual substring of the entry to be annotated.

```
<!-- ... -->
<text>
  Today I had a meeting with
  <annotation>
    <mainAnchor>http://sw.deri.org/~aharth</mainAnchor>
    <metadata>
      <foaf:Person>
        <foaf:name>Andreas Harth</foaf:name>
        <!-- ... other properties -->
      </foaf:Person>
    </metadata>
    <content>Andreas</content>
  </annotation>.
  We went over his
  <annotation>
    <mainAnchor>
      http://sw.deri.org/2004/06/yars/doc/summary
    </mainAnchor>
```

```
      <metadata>
        <bibtex:InProceedings>
          <bibtex:title>
            Yet Another RDF Store:...
          </bibtex:title>
          <!-- ... other properties -->
        </bibtex:InProceedings>
      </metadata>
      <content>paper</content>
    </annotation>
    and talked about possibilities of using his YARS RDF
    store for a manned mission to Alpha Centauri. ...
  </text>
  <!-- ... -->
```

Fig. 4. Intermediate XML

XSLT Transformation

Once the user's textual entry and the semantic metadata have been combined into the intermediate XML format, we can use XSLT to transform it into arbitrary publication channels. This is a rather straight-forward process: there is one XSLT stylesheet per channel, and each one picks those elements out of the XML, which are appropriate for the corresponding publication channel. For example, the XML → HTML stylesheet will ignore the `metadata` element and instead pick the `content` and `mainAnchor` tags to produce a hyperlink for specific substrings of the entry. The XML → RSS stylesheet will only look at the `metadata` element and annotate the entry as a whole. Once this is done, the blog (now consisting of various files) can be uploaded to a server.

Future Work

Although the full chain of steps involved in publishing a semantic blog – writing a textual entry, adding pictures, annotating the entry with semantic metadata and transforming the internal data format into the actual components of the blog – is implemented and can be performed within semiBlog, the software is still very much a prototype and can be improved in many areas. Since the main focus was on the interaction between semiBlog and other desktop applications, we have so far neglected aspects such as layout capabilities. As a result, the user is currently bound to a static layout scheme for the blog. In future versions, we will have to address this and perhaps allow the user to choose between different Cascading Style Sheets (CSS) templates. Also, it is currently not possible to add metadata other than that provided by the wrapper plug-ins. Users should have the option to manually annotate entries with arbitrary metadata, if they wish to do so

(however, since we were interested in *easy* drag-and-drop based annotation, this was of no concern to us for the prototype). Also, by using an ontology like SIOC [2], more semantic metadata could be generated easily by making use of the internal semantic structure of the blog.

Future versions of the editor will allow integration with generic Semantic Desktop solutions like Gnowsis [13]. This would eliminate the need for application specific data wrappers as they are currently used in semiBlog.

Conclusion

In this paper, we have shown that the recent phenomenon of blogging, combined with a tool to easily generate SW data from existing formal desktop data, can result in a form of semantic blogging. We discussed various ways of representing semantic metadata in a blog and decided to apply indirect annotation through RSS 1.0 feeds. We presented a prototype of the *semiBlog* editor, which was created with the purpose of semantic blogging in mind. We argued that such a semantic blog editor should integrate tightly with a user's desktop environment, as this allows direct API access to external applications and user interface idioms such as drag-and-drop and makes integration of existing data into the blog as easy as possible. We believe that by providing such a tool, we will increase the motivation to add to and to help realise the vision of the Semantic Web.

Acknowledgement

This material is based upon works supported by the Science Foundation Ireland under Grant No. SFI/02/CE1/I131.

References

[1] Beged-Dov G, Brickley D, Dornfest R, Davis I, Dodds L, Eisenzopf J, Galbraith D, Guha R, MacLeod K, Miller E, Swartz A, van der Vlist E (2001) RDF Site Summary (RSS) 1.0. http://web.resource.org/rss/ 1.0/spec

[2] Breslin JG, Harth A, Bojars U, Decker S (2005) Towards Semantically Interlinked Online Communities. Proc of the 2nd European Semantic Web Conference (ESWC '05) Heraklion Greece May 2005

[3] Brickley D, Miller L (2005) FOAF Vocabulary Specification. http://xmlns.com/foaf/0.1/.

[4] Cayzer S (2004) Semantic Blogging: Spreading the Semantic Web Meme. Proc of XML Europe 2004 Amsterdam Netherlands April 2004
[5] Haase P, Broekstra J, Ehrig M, Menken M, Mika P, Plechawski M, Pyszlak P, Schnizler B, Siebes R, Staab S, Tempich C (2004) Bibster - A Semantics-Based Bibliographic Peer-to-Peer System. Proc of the Third International Semantic Web Conference (ISWC2004) Hiroshima Japan November 2004
[6] Hendler J (2001) Agents and the Semantic Web. IEEE Intelligent Systems, vol 16, no 2, pp 30–37
[7] Karger DR, Quan D (2004) What Would It Mean to Blog on the Semantic Web? Proc of the Third International Semantic Web Conference (ISWC2004) Hiroshima Japan November 2004
[8] Ohmukai I, Takeda H (2004) Semblog: Personal Publishing Platform with RSS and FOAF. Proc of the 1st Workshop on Friend of a Friend Social Networking and the (Semantic) Web Galway September 2004
[9] Palmer SB (2002) RDF in HTML: Approaches. http://infomesh.net/ 2002/rdfinhtml/.
[10] Pilgrim M (2002) What is RSS? www.xml.com/pub/a/2002/ 12/18/dive-into-xml.html.
[11] Quan D, Bakshi K, Karger DR (2003) A Unified Abstraction for Messaging on the Semantic Web. Proc of the Twelfth International World Wide Web Conference (WWW2003) Budapest Hungary May 2003
[12] Quan D, Huynh D, Karger DR (2003) Haystack: A Platform for Authoring End User Semantic Web Applications. Proc of the Second International Semantic Web Conference (ISWC2003)
[13] Sauermann L (2003) The Gnowsis — Using Semantic Web Technologies to build a Semantic Desktop. Master's Thesis Technische Universität Wien
[14] Walker J (2005) Weblog. In: Routledge Encyclopedia of Narrative Theory. Routledge London and New York