LibSwb: Browsing the Entity Context*

Sven Buschbeck, Stefano Bortoli, and Heiko Stoermer

University of Trento,
Dept. of Information Science and Engineering,
Trento, Italy
{buschbeck, bortoli, stoermer}@disi.unitn.it

Abstract. In this paper we are presenting ongoing work on a software library for on-the-fly browsing of Semantic Web content that forms the context around an entity. This library is targeted to be integrated into user applications, to extend their range of functionalities in an easy-to-handle and unobtrusive way. We illustrate an envisioned application, give details on the implementation of a first prototype, and evaluate its feasibility.

Keywords: Semantic Web browsing, entity context, software library, on-the-fly, RDF

1 Introduction

One of the most important differences between the Semantic Web and traditional Knowledge Representation is the concept of the URI as the global anchor for information about a resource in distributed information sources. This anchor should – in theory – allow us to gather, merge and prepare for consumption distributed fragments of RDF (and OWL) content, potentially applying filtering techniques, contextual representation, and restructuring either for a human or machine agent.

A serious complication in this respect is the fact that in its current state, the Semantic Web does not necessarily feature any linking as the WWW does, in the form of hyperlinks that can be dereferenced and lead from document to document: a URI for a resource can be completely arbitrary, and does not need to expose any reference to actual data about the resource. A second issue is the uncoordinated way in which URIs are minted, a problem that has extensively been discussed in several papers [7, 6], and that leads to a proliferation of URIs for the same object.

The ongoing work we are describing in this paper arises from requirements of the EU-funded project OKKAM, in which one of the objectives is to enable what we call “entity-centric browsing” of (Semantic) Web content. The main objective of the OKKAM project is the creation of an Entity Name System (ENS),

* This work is partially supported by the FP7 EU Large-scale Integrating Project OKKAM – Enabling a Web of Entities (contract no. ICT-215032). For more details, visit http://www.okkam.org.
a global, public service for the systematic creation and re-use of identifiers for entities (see [7] for more details). The paradigm we are trying to promote and explore is that – unlike in the WWW today – at the center of attention we do not place documents, but entities, i.e. people, events, locations, organizations, and more. Starting from an entity, we want to enable users to browse data in an innovative way, using the entity’s URI as a pivot for semantic data (the “entity context”), which we gather from the distributed sources that have something to say about this entity.

This entity-centric approach coincides with the objectives of the Semantic Web, in that the RDF data model basically consists of binary predicates about “something”. We want to bring this approach to a next level, by providing a new layer of navigation on top of the traditional web: entity centric browsing as an addition to document-centric browsing.

To illustrate this paradigm, in Sect. 2 we sketch out an application mockup that we are envisioning and which explains our goals in more detail. While Sect. 3 discusses work related to our problem, Sect. 4 describes the core work presented in this paper: our first prototype of LibSwb\(^1\), which provides the functionality of entity-centric data collection from distributed sources. First evaluations to illustrate the feasibility of the approach are given in Sect. 5. The article closes with a discussion of future work (Sect. 6) and conclusion.

## 2 Motivation

The drive to start working on LibSwb is the need to display what we call the semantic “entity context” in a browser.

The idea is to define a plugin for a browser, enabling the combination of World Wide Web and Semantic Web information, thus triggering an exploration of an entity-centric knowledge space. A sketch of a possible workflow of this application is following:

- The user selects a set of keywords identifying an entity. Those keywords are used to inquire the ENS in order to retrieve the entity URI. This URI is used as pivot URI in LibSwb starting the Semantic Web browsing process (see Fig. 1).
- All the RDF documents retrieved are merged and the contained information is used to extend the context menu related to the pivot entity URI. The user now is enabled to explore the context menu following the thread of information of his/her interest as described in figure 1.
- If the user is interested, he/she can switch on graph browsing, starting a “parallel” Semantic Web content exploration. Once the user finds a resource particularly interesting, he/she is enabled to visualize a list of HTTP URLs pointing at documents containing references to the considered entity, if any (see Fig. 1). The selection of one of these URLs simply triggers the opening of another browser page presenting its content and switching the browsing back to the WWW.
The important fact to note is that neither do data displayed in the entity context need to be manually maintained, nor do originate from the same data source. Instead, they are automatically retrieved, joined and displayed, on-the-fly, once a pivot URI has either been provided from an annotation in a retrieved document, or through entity search in the ENS.

Differently from the World Wide Web, the Semantic Web relies on virtual/logical references connecting information about entities (or resources) contained in distributed and independent documents. These references are URIs and inverse-functional property’s values. Unlike WWW references connecting hypertexts defined in different documents (i.e. HTML anchor links), in the Semantic Web different URIs can be used to refer to the same entity (or resource).

1 Pronounced “libswib”, an acronym for LIBrary for Semantic Web Browsing
When URIs refer to the same entity, this logical equivalence is stated by means of “owl:sameAs” statements [1, 4, 2]. The weak identification approach currently adopted in the Semantic Web, combined with the cross references in physically distributed documents, makes on-the-fly browsing a challenging activity. In order to perform on-the-fly entity-centric browsing it is necessary to perform several complex operations which we have laid out in Sect. 4.

3 Related Work

In this section, four software projects will be discussed. They all present, to a certain extent, solutions to the Semantic Web browsing topic.

Piggy Bank[8] is a plugin for the Mozilla Firefox browser presented by David Huynh, Stefano Mazzocchi and David Karger. This plugin collects semantically annotated data from websites. Users can tag content and build tag ontologies called folksologies. The goal is to make the Semantic Web easily usable and at the same time build up information repositories. The structured repository content is shared among the users increasing the available Semantic Web knowledge. However, a conflict of interests between content providers and the consumers is not peered enough. The authors assume that content providers will be motivated in releasing their content in proper RDF, because otherwise “screen scrapers” will automatically collect information by parsing provider’s HTML pages and therefore the control over information is off the provider’s hands. However, it is questionable whether commercial content providers are interested in sharing the “pure” information, as this would make their offers be easily comparable to competitors and therefore could be a disadvantage in terms of marketing issues. LibSwb instead is intended to be used within any kind of application. Thus, any provider is free to create software fitting its needs.

By default, Piggy Bank either stores the gathered information locally, disconnecting it from its origin, or requires on demand information reload. Notice that in case of on demand reloading, if the resource has been moved in another location, the data are lost. Relying on search engines as Sindice [15], LibSwb collects always “fresh” information avoiding any loss of data. Furthermore, LibSwb could also use the information shared in the repositories of Piggy Bank/Semantic Bank.

A further advantage of the LibSwb approach is that any information is retrieved automatically from distributed resources whereas any Piggy Bank user

---

2 [http://www.mozilla.com/firefox](http://www.mozilla.com/firefox)

3 Tags are modeled as RDF resources with URIs so that they can be annotated and thereby grouped and hierarchically ordered.

4 “Screen scrapers” are pieces of software mainly written by Piggy Bank users and they are able to extract information by parsing HTML pages. This information is then converted to RDF, representing the scraper writer’s point of view about the information’s semantics.

5 Semantic Bank is the name of the server-side software to share collected RDF data among the Piggy Bank users.
needs to connect explicitly to a Semantic Bank to retrieve updates or details about the stored information.

**Haystack** [11] is a standalone Semantic Web browser application, presented by Dennis Quan, capable of visualizing RDF documents using VOWL (View Ontology Web Language). A claim of the author is that “One finds many examples of a user wishing to see the same underlying collection\(^6\) in different ways”. This claim, combined with the release of Haystack as a standalone browser, seems to assume that the web only consists of collections. However, it is possible to observe that the web actually contains a substantial amount of plain text to convey any kind of message than plain information. LibSwb, as a part of a web application, can provide Semantic Web access functionalities when needed. For example, it can be used as an extension of a web browser rather than as a replacement of it. Additionally, by providing an extensive range of plugins, a web browser like Mozilla Firefox fulfills much more functions than just browsing the web, which will keep sophisticated users from switching to Haystack.

Proposed starting points for Haystack are either a bookmark or a ‘Go to box’ where the user is supposed to enter a URL. LibSwb has the advantage of being enabled to retrieve URIs or okkam identifiers automatically by starting from a set of keywords, which are usually easier to handle and to remember.

**Potluck** [9], created by David F. Huynh, Robert C. Miller and David R. Karg, is a web-application meant to help Internet users creating their own information mash-ups starting from distributed data sources, without any coding skills or Semantic Web knowledge. Even though there is definitely a need for software helping users performing these kind of tasks, Potluck relies on the user to know, how and where to find URLs of RDF documents, that can be combined. LibSwb is intended to be a generic solution to either find appropriate resources about a specific topic or to extend known ones with sources “talking” about the same (or a related) topic.

**Tabulator** [14] is a Semantic Web browser defined in the context of the LinkedData initiative [2, 5]. Similarly to LibSwb, Tabulator aims for on-the-fly Semantic Web browsing. The main difference in the approach is indeed, browsing operations are performed by following property values and URIs as links. Tabulator relies on the LinkedData reference resolution system based on HTTP protocol [12, 4], enabling browsing from one RDF document to another. This browsing system fails if one of the browsing documents does not present any reference to further RDF documents. LibSwb however takes this idea of a minimalistic approach but enriches the results with information discovered by the use of inverse functional properties or by tracking down virtually referenced resources. In the end, Tabulator is conceived to browse RDF documents, whereas LibSwb is conceived to browse the “entity context” distributed in RDF documents spread on the Semantic Web.

\(^6\) A collection is meant as a set of information in any order or structure
4 Solution Approach

This section aims at presenting the solution approach adopted tackling the entity-centric Semantic Web browsing as outlined in Sect. 2. The idea behind our work is to enable Semantic Web browsing, replicating the behavior of a WWW browser hiding the complexity of the ongoing operations. Namely, Semantic Web browsing should consist in moving through the “Giant Global Graph” [3] by presenting a bounded view (or subgraph) focusing on one entity pivot at each browsing step. The operation that triggers the browsing step is simply the selection of an entity as pivot point. The solution we propose is a software library called LibSwb, implementing, in a simple software interface, the underlying browsing operations.

A brief description of the solutions chosen for each of the operations necessary to realize such a browsing process is presented in the following.

**State starting pivot URI.** The definition of an entry point is an essential operation to trigger any kind of browsing. The solution we propose to fill this task is simply to allow the submission of an URI used as start pivot. Furthermore, we propose an alternative starting point based on search results of the ENS. Namely, the Semantic Web browsing can start by searching the pivot URI through the ENS API (e.g. keyword based search). The entity URI result of the search is then used as entry point, starting the browsing activities.

**Retrieve pivot URI alternatives.** This task is essential to enable an extensive entity related “information integration”. Our solution relies on the ENS to gather this kind of “alternative identifiers” and enhance the mining of information related to the resource on the center of the browsing activity.

**Retrieve information about the pivot URI.** The approach adopted to solve this task relies on the existence of Semantic Web search engines and indexes. Basically, this operation consists in inquiring search engines as Sindice [15] and Falcon [10] looking for RDF documents including the pivot URI and its alternatives. Notice that using URIs to inquire search engine guarantees an 100% precision in the returned result as long as no similarity analysis is performed as for keyword based search. Furthermore, our solution approach is not constrained with using only one search engine. The use of several searching sources guarantees a wider coverage of Semantic Web browsable content. It is important to notice how the proliferation of pivot URI’s alternative identifiers can affect this operation. Indeed, Semantic Web search should be repeated for any of the alternatives URI creating a search overhead affecting browsing performances. An extensive use of OKKAM to define and share entity URIs would prevent this searching extrawork.

**Building an “artifact” RDF graph.** This task is the most expensive in terms of computational costs. The merging and pruning of RDF graphs can heavily affect the performances of on-the-fly browsing when it involves automatic reasoning on large set of resources. Also in this case it is important to notice how this process would benefit of an extensive use of OKKAM as source of entity identifiers. Indeed, in an ideal world, if any entity would be identified by a unique
URI across all the Semantic Web, the RDF graph merging operation would come almost for free, enhancing browsing performances.

The aim of the Java implementation of a LibSwb prototype is oriented towards defining a “proof of concept”, allowing to perform some experiments about Semantic Web browsing (see section 5). The prototype relies on the Jena framework\(^7\) for RDF graph management, the Sindice Semantic Web Index\(^8\) for searching the Semantic Web, and the ENS to actually find a pivot URI and retrieving alternative URIs related to it.

The library mainly exposes two public methods:

- \texttt{LibSwb.init(string:startPivotUri)} this method is the entry point for starting browsing the Semantic Web. The startURI is used to inquire Sindice and retrieve a first set of RDF documents initializing the generation of the very first “artifact graph”.
- \texttt{LibSwb.changePivotUri(string:newPivotUri)} this method triggers the browsing of the Semantic Web, moving the focus from the former pivot URI to the new one.

The execution of both methods trigger a query to Sindice asking for the RDF documents containing the pivot URI. The list of document is then processed forming the “artifact graph” returned as result. The download and parsing of the RDF document is performed through the Jena API, which returns a Jena Model 'object'. This RDF graph is then pruned of the irrelevant information\(^9\) and merged with the “artifact graph” under construction. The merging of RDF graphs is performed using the Jena API functionalities\(^10\). The pruning of an RDF graph is a way more complex, based on Breadth First Search algorithm\(^11\). Basically the algorithm works by 'walking' the RDF graph collecting all the nodes that are neighbors of the 'pivot' within certain distance threshold. The list of nodes is then used to retrieve all the statements that are going to form and extend the “artifact graph”. The pseudo-code of the pruning algorithm is presented in Algorithm 1.

This version of the prototype implements also a method allowing to trigger the browsing process starting from a set of keywords. This feature relies on okkam’\textquoteleft s capability in retrieving identifiers of resources given a keyword based description.

An evaluation with this first implementation of LibSwb is presented in section 5.

\(^7\) http://jena.sourceforge.net/
\(^8\) http://sindice.com/
\(^9\) We define as irrelevant all the RDF statements involving nodes that are beyond a threshold distance from the pivot node.
\(^10\) The operation logically performs the ‘union’ of the RDF Graphs by collapsing nodes identified by the same URI ‘wrapping’ all related RDF statements.
\(^11\) Computational time complexity can be estimated with worst case bound to $O(|E| + |V|)$ where $E$ is the set of RDF statements (graph edges) and $V$ is set by resources considered. Notice that this pruning algorithm always considers only a finite set of nodes within a certain threshold distance, avoiding infinite divergence.
**Algorithm 1** The graph pruning algorithm

```java
prune(Graph graph, String pivotURI, int maxDepth)
List visited = [];
Node node = new Node(level=0,pivotURI);
queue.enqueue(node);
while queue.hasItems & n = queue.dequeue() do
  if n.depth < maxDepth then
    Node[] list = g.getNeighbors(n);
    while (list.hasItems & neighbor = list.next()) do
      if ! visited.contains(neighbor) then
        neighbor.setLevel+1; queue.enqueue(neighbor); visited.add(neighbor);
      end if
    end while
  end if
end while
Graph newGraph;
while visited.hasItems do
  Node n = visited.next(); newGraph = newGraph ∪ (g.selectStmt(n));
end while
return newGraph
```

5 Evaluation

The most critical part of browsing hypertext documents is the time, a system is occupied with downloading a web-resource and visualize it. This procedure should optimally be fulfilled in real-time to avoid the user canceling the process due to the limited attention span. The LibSwb approach is also bound to file download but additionally includes more complex procedures as described in section 4 and therefore higher computational costs will be due.

The test scenario chosen reflects an intended average use of LibSwb and will be explained in detail as follows. The assumed intention is to browse the social environment of Sven Buschbeck, therefore LibSwb is initialized with “Sven Buschbeck” as keywords. All retrieved files are loaded and merged together to form the first RDF representation of Sven Buschbeck.

As next step, the focus is moved on Stefano Bortoli as he is mentioned as "known" person in the retrieved FOAF profile of Sven Buschbeck. This change of the pivot point leads to an extension of the existing RDF graph with information contained in all the files presenting the OKKAM ID of Stefano Bortoli. To keep the point of interest focused, the graph is pruned to the first level of referenced nodes as described in section 4.

This step is repeated by changing the pivot point to Heiko Stoermer referred to as “known” in Stefano’s FOAF profile leading to another extension followed by pruning of the graph.

By the steps described above 11 documents with more than 300 statements in 60kB of RDF data have been merged together as shown in table 1.
merged documents 11
average document size [in byte] 5636
total size of documents [in byte] 61992
average number of RDF statements per document 28
total number of RDF statements in all documents 310
number of statements in merged document 172
number of resource descriptions in merged document 53

Table 1. Benchmark data of the test scenario.

The test procedure mentioned has been performed one hundred times on three different machines\textsuperscript{12} and led to runtimes as shown in table 2.

<table>
<thead>
<tr>
<th>system</th>
<th>total for downloading [in ms]</th>
<th>computation [in ms]</th>
<th>average per step [in ms]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pentium4 3GHz</td>
<td>6847.42</td>
<td>6827.43</td>
<td>19.99</td>
</tr>
<tr>
<td>Centrino 1.8GHz</td>
<td>11323.98</td>
<td>10198.73</td>
<td>1125.25</td>
</tr>
<tr>
<td>Centrino 2.4GHz</td>
<td>17641.27</td>
<td>14788.71</td>
<td>2852.56</td>
</tr>
</tbody>
</table>

Table 2. Runtimes obtained by running the same test scenarios on three different machines in three different geographical locations.

The test reveals the lack of a multi-threading architecture which will be considered in a follow up version. However it has been shown, that our on-the-fly Semantic Web browsing approach is feasible with a yet simple prototype as every step of the test procedure has been fulfilled below one second. For every optimization in future versions, more sophisticated and complex algorithms can be used still producing an instant result.

6 Future Work

As previously stated, the work described in this article is in an early stage of development. In this section we present some of the topics we have scheduled for the next months.

An important aspect which we will consider is contextuality: representation and management of the data retrieved about an entity. Context-sensitivity of RDF data has been addressed on several levels\textsuperscript{13}, and has found its way into several RDF management systems and libraries. We are going to extend LibSwb in this direction, to lay the foundations for several context-related aspects, including

\textsuperscript{12} Machines used: “Pentium4 3GHz”: Intel\textsuperscript{®} Pentium\textsuperscript{TM} 4 @ 3GHz CPU, 3GB RAM, “Centrino 1.8GHz”: Intel\textsuperscript{®} Core\textsuperscript{TM} 2 Duo T5600 @ 1.83GHz CPU, 2GB RAM, “Centrino 2.4GHz”: Intel\textsuperscript{®} Core\textsuperscript{TM} 2 Duo T7700 @ 2.4GHz CPU, 2GB RAM

\textsuperscript{13} We refer the reader to [13] and the contained Related Work section.
the limitations of graph merging, provenance and quality of data, as well as contextual visual representation of information.

The current version of the LibSwb prototype relies only on Sindice to retrieve RDF documents, ignoring further browsing information contained in the RDF document itself (e.g. rdf:seeAlso statements), or available through other indexing services. The next evolution of the library will also consider the references to further RDF data embedded in documents or retrieved through Semantic Web search. In this way, we are moving in the direction vision of the W3C Semantic Web research group about RDF browsing (i.e. Tabulator\textsuperscript{14} RDF Browser). To accommodate these changes, it is necessary to frame the current implementation in a pluggable architecture, allowing the integration of different Semantic Web data sources such as Falcon and others \cite{10, 15}.

Finally, we are observing a trend in the diffusion of microformats\textsuperscript{15} to define semantically structured data embedded in HTML documents. Microformats are open data formats designed more for humans than machines, and generally built upon existing and widely adopted standards. The “human-oriented” less formal design of microformats might ease its diffusion, for this reason it could be interesting to explore a way to integrate also this kind of information in the Semantic Web browsing activities.

7 Conclusion

This paper presents a first step of an ongoing research with the aim of enabling entity-centric Semantic Web browsing. LibSwb is intended to be used within many kinds of applications that require a convenient interface to Semantic Web content, hiding the difficulties of how to find and retrieve such content, and being extensible to accommodate new data sources and services.

We have described a problem setting and the very specific requirement of providing a way to gather distributed RDF data, in order to reveal the “semantic context” of an entity. Of course this is transferable to different kinds of applications, e.g. authoring systems that need to provide up-to-date background information about entities, or extensions to web browsers that add a navigational layer on top of document hyperlink structures.

We hope that this can ease the development of a range of entity-centric (Semantic) Web enabled applications, in the context of the OKKAM project, but of course also in the larger community. Our declared goal is to extend the functional range of networked applications, and to promote the paradigm entity-centric access to information. We are thus planning to make the library publicly available as soon as the most important features mentioned in the previous section are implemented and sufficiently stable.

\textsuperscript{14} http://www.w3.org/2005/ajar/tab
\textsuperscript{15} http://microformats.org/
References


