Available online at www.sciencedirect.com
ScienceDirect
Web Semantics: Science, Services and Agents on the World Wide Web xxx (2007) xxx–xxx
JOURNAL OF Web Semantics
www.elsevier.com/locate/websem

Piggy Bank: Experience the Semantic Web inside your web browser

David Huynh [a,*], Stefano Mazzocchi [b], David Karger [a]

[a] *MIT Computer Science and Artificial Intelligence Laboratory, The Stata Center, Building 32, 32 Vassar Street, Cambridge, MA 02139, USA*
[b] *MIT Digital Libraries Research Group, 77 Massachusetts Ave., Cambridge, MA 02139, USA*

Received 10 June 2006; received in revised form 15 December 2006; accepted 15 December 2006

Abstract

The Semantic Web Initiative envisions a Web wherein information is offered free of presentation, allowing more effective exchange and mixing across web sites and across web pages. But without substantial Semantic Web content, few tools will be written to consume it; without many such tools, there is little appeal to publish Semantic Web content.

To break this chicken-and-egg problem, thus enabling more flexible information access, we have created a web browser extension called *Piggy Bank* that lets users make use of Semantic Web content within Web content as users browse the Web. Wherever Semantic Web content is not available, Piggy Bank can invoke screenscrapers to re-structure information within web pages into Semantic Web format. Through the use of Semantic Web technologies, Piggy Bank provides direct, immediate benefits to users in their use of the existing Web. Thus, the existence of even just a few Semantic Web-enabled sites or a few scrapers already benefits users. Piggy Bank thereby offers an easy, incremental upgrade path to users without requiring a wholesale adoption of the Semantic Web's vision.

To further improve this Semantic Web experience, we have created *Semantic Bank*, a web server application that lets Piggy Bank users share the Semantic Web information they have collected, enabling collaborative efforts to build sophisticated Semantic Web information repositories through simple, everyday's use of Piggy Bank.

Keywords: Semantic Web; Screen scraping; Browsing; Sharing; Tagging; Annotation; Collaboration

1. Introduction

The World Wide Web has liberated information from its physical containers—books, journals, magazines, newspapers, etc. No longer physically bound, information can flow faster and more independently, leading to tremendous progress in information usage.

But just as the earliest automobiles looked like horse carriages, reflecting outdated assumptions about the way they would be used, information resources on the Web still resemble their physical predecessors. Although much information is already in structured form inside databases on the Web, such information is still flattened out for presentation, segmented into "pages," and aggregated into separate "sites." Anyone wishing to retain a piece of that information (originally a structured database record) must instead bookmark the entire containing page and continuously repeat the effort of locating that piece within the page. To collect several items spread across multiple sites together, one must bookmark all of the corresponding containing pages. But such actions record only the pages' URLs, not the items' structures. Though bookmarked, these items cannot be viewed together or organized by whichever properties they might share.

Search engines were invented to break down web sites' barriers, letting users query the whole Web rather than multiple sites separately. However, as search engines cannot access to the structured databases within web sites, they can only offer unstructured, text-based search. So while each site (e.g., epicurious.com) can offer sophisticated structured browsing and searching experience, that experience ends at the boundary of the site, beyond which the structures of the data within that site is lost.

In parallel, screenscrapers were invented to extract fragments within web pages (e.g., weather forecasts, stockquotes, and news article summaries) and re-purpose them in person-

* Corresponding author. Tel.: +1 617 733 3647.
E-mail addresses: dfhuynh@csail.mit.edu (D. Huynh), stefanom@mit.edu (S. Mazzocchi), karger@csail.mit.edu (D. Karger).

1570-8268/$ – see front matter © 2007 Published by Elsevier B.V.
doi:10.1016/j.websem.2006.12.002

alized ways. However, until now, there is no system in which different screenscrapers can pool their efforts together to create a richer, multi-domained information environment for the user.

On the publishing front, individuals wishing to share structured information through the Web must think in terms of a substantial publication process in which their information must be carefully organized and formatted for reading and browsing by others. While Web logs, or blogs, enable lightweight authoring and have become tremendously popular, they support only unstructured content. As an example of their limitation, one cannot blog a list of recipes and support rich browsing experience based on the contained ingredients.

The Semantic Web [22] holds out a different vision, that of information laid bare so that it can be collected, manipulated, and annotated independent of its location or presentation formatting. While the Semantic Web promises much more effective access to information, it has faced a chicken-and-egg problem getting off the ground. Without substantial quantities of data available in Semantic Web form, users cannot benefit from tools that work directly with information rather than pages, and Semantic Web-based software agents have little data to show their usefulness. Without such tools and agents, people continue to seek information using the existing web browsers. As such, content providers see no immediate benefit in offering information natively in Semantic Web form.

1.1. Approach

In this paper, we propose *Piggy Bank*, a tool integrated into the contemporary web browser that lets Web users extract individual information items from within web pages and save them in Semantic Web format (RDF [20]), replete with metadata. Piggy Bank then lets users make use of these items right inside the same web browser. These items, collected from different sites, can now be browsed, searched, sorted, and organized together, regardless of their origins and types. Piggy Bank's use of Semantic Web technologies offers direct, immediate benefits to Web users in their everyday's use of the existing Web while incurring little cost on them.

By extending the current web browser rather than replacing it, we have taken an incremental deployment path. Piggy Bank does not degrade the user's experience of the Web, but it can improve their experience on RDF-enabled web sites. As a consequence, we expect that more web sites will see value in publishing RDF as more users adopt Piggy Bank. On sites that do not publish RDF, Piggy Bank can invoke screenscrapers to re-structure information within their web pages into RDF. Our two-prong approach lets users enjoy however few or many RDF-enabled sites on the Web while still improving their experience on the scrapable sites. This solution is thus not subject to the chicken-and-egg problem that the Semantic Web has been facing.

To take our users' Semantic Web experience further, we have created Semantic Bank, a communal repository of RDF to which a community of Piggy Bank users can contribute to share the information they have collected. Through Semantic Bank, we introduce a mechanism for lightweight structured information publishing and envision collaborative scenarios made possible by this mechanism.

Together, Piggy Bank and Semantic Bank pave an easy, incremental path for ordinary Web users to migrate to the Semantic Web while still remaining in the comfort zone of their current Web browsing experience.

2. User experience

First, we describe our system in terms of how a user, Alice, might experience it for the task of collecting information on a particular topic. Then we extend the experience further to include how she shares her collected information with her research group.

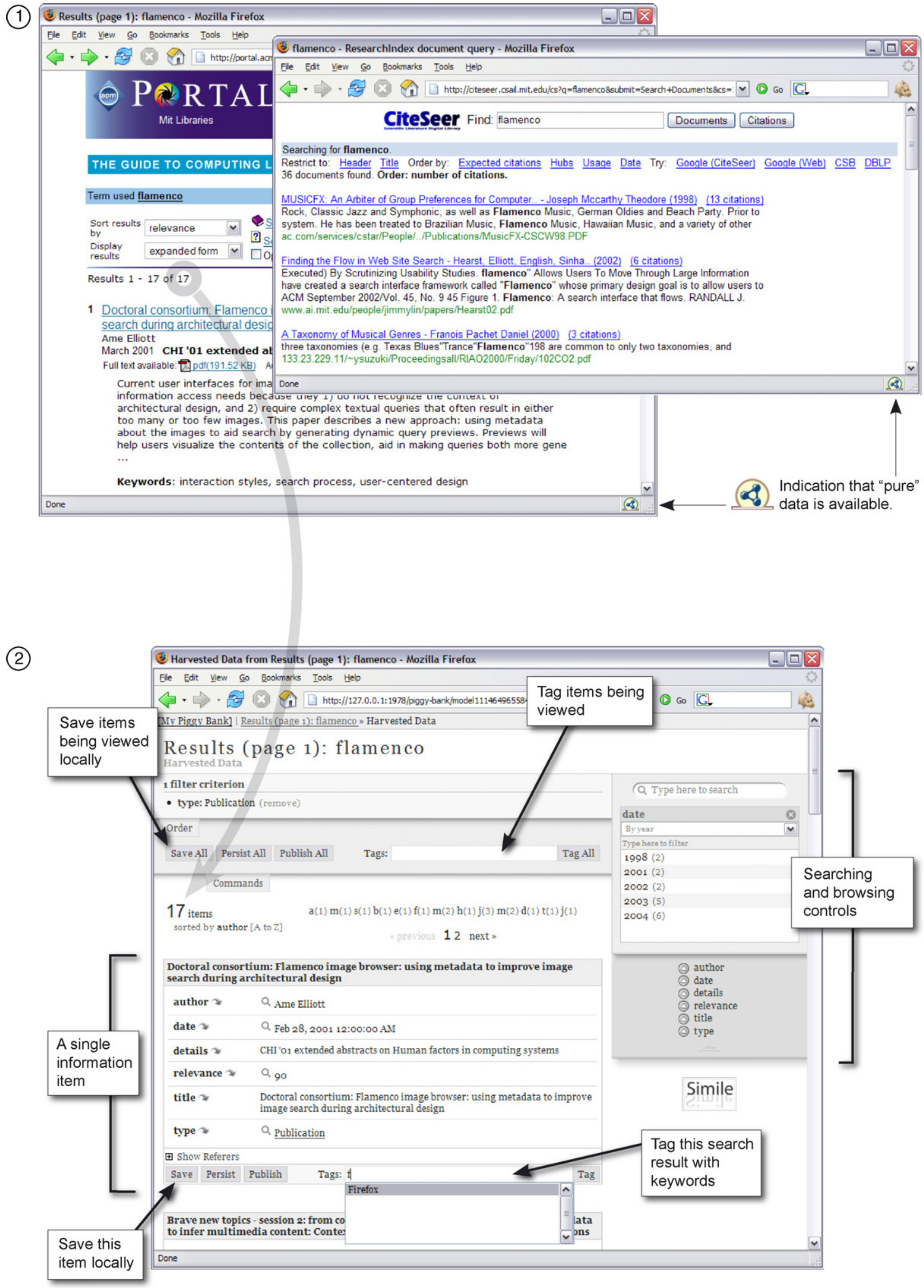
2.1. Collecting information

Alice searches several web sites that archive scientific publications (Fig. 1). The *Piggy Bank* extension in Alice's web browser shows a "data coin" icon in the status bar for each site, indicating that it can retrieve the same information items in a "purer" form. Alice clicks on that icon to collect the "pure" information from each web site. In Fig. 2, Piggy Bank shows the information items it has collected from one of the sites, right inside the same browser window. Using Piggy Bank's browsing facilities, Alice pinpoints a few items of interest and clicks the corresponding "Save" buttons to save them locally. She can also tag an item with one or more keywords, e.g., the topic of her search, to help her find it later. The "tag completion" dropdown suggests previously used tags that Alice can pick from. She can also tag or save several items together.

Alice then browses to several RSS-enabled sites from which she follows the same steps to collect the news articles relevant to her research. She also 'googles' to discover resources that those publication-specific sites do not offer. She browses to each promising search result and uses Piggy Bank to tag that web page with keywords (Fig. 3).

After saving and tagging several publications, RSS news articles, and web pages, Alice browses to the local information repository called "My Piggy Bank" where her saved data resides (Fig. 4). She clicks on a keyword she has used to tag the collected items (Fig. 4) and views them together regardless of their types and origins (Fig. 5). She can sort them all together by date to understand the overall progress made in her research topic over time, regardless of how the literature is spread across the Web.

Now that the information items Alice needs are all on her computer, rather than being spread across different web sites, it is easier for her to manage and organize them to suit her needs and preferences. Throughout this scenario, Alice does not need to perform any copy-and-paste operation, or re-type any piece of data. All she has to do is click "Save" on the items she cared about and/or assign keywords to them. She does not have to switch to a different application—all interactions are carried out within her web browser which she is already familiar



Figs. 1–2. (1) The Piggy Bank extension to the web browser indicates that it can "purify" data on various websites. (2) Piggy Bank shows the "pure" information items retrieved from ACM.org. These items can be refined further to the desired ones, which can then be saved locally and tagged with keywords for more effective retrieval in the future.

Please cite this article in press as: D. Huynh et al., Piggy Bank: Experience the Semantic Web inside your web browser, Web Semantics: Sci. Services Agents World Wide Web (2007), doi:10.1016/j.websem.2006.12.002

Fig. 3. Like del.icio.us, Piggy Bank allows each web page to be tagged with keywords. However, this same tagging mechanism also works for "pure" information items and is indiscriminate against levels of granularity of the information being tagged.

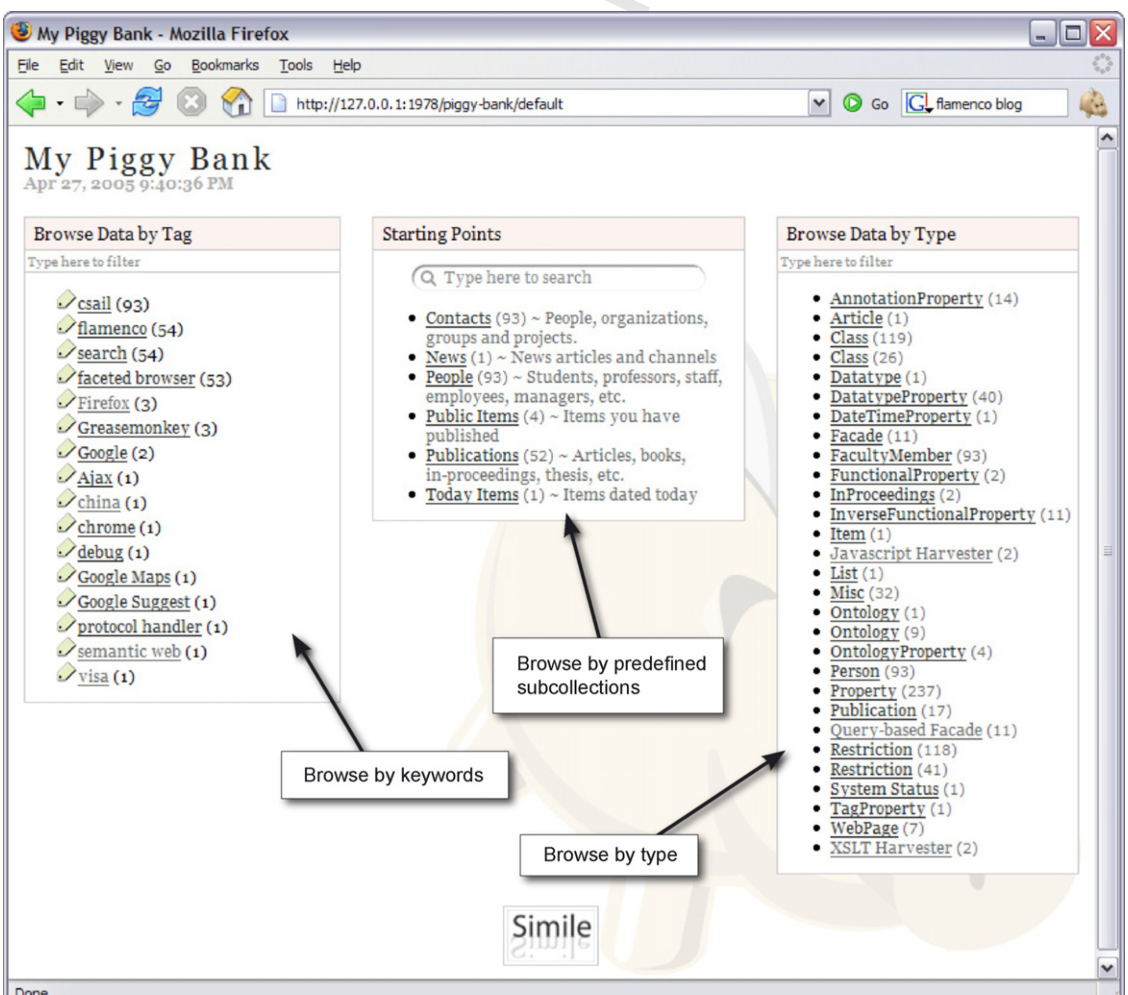


Fig. 4. Saved information items reside in "My Piggy Bank." The user can start browsing them in several ways, increasing the chances of re-finding information.

Please cite this article in press as: D. Huynh et al., Piggy Bank: Experience the Semantic Web inside your web browser, Web Semantics: Sci. Services Agents World Wide Web (2007), doi:10.1016/j.websem.2006.12.002

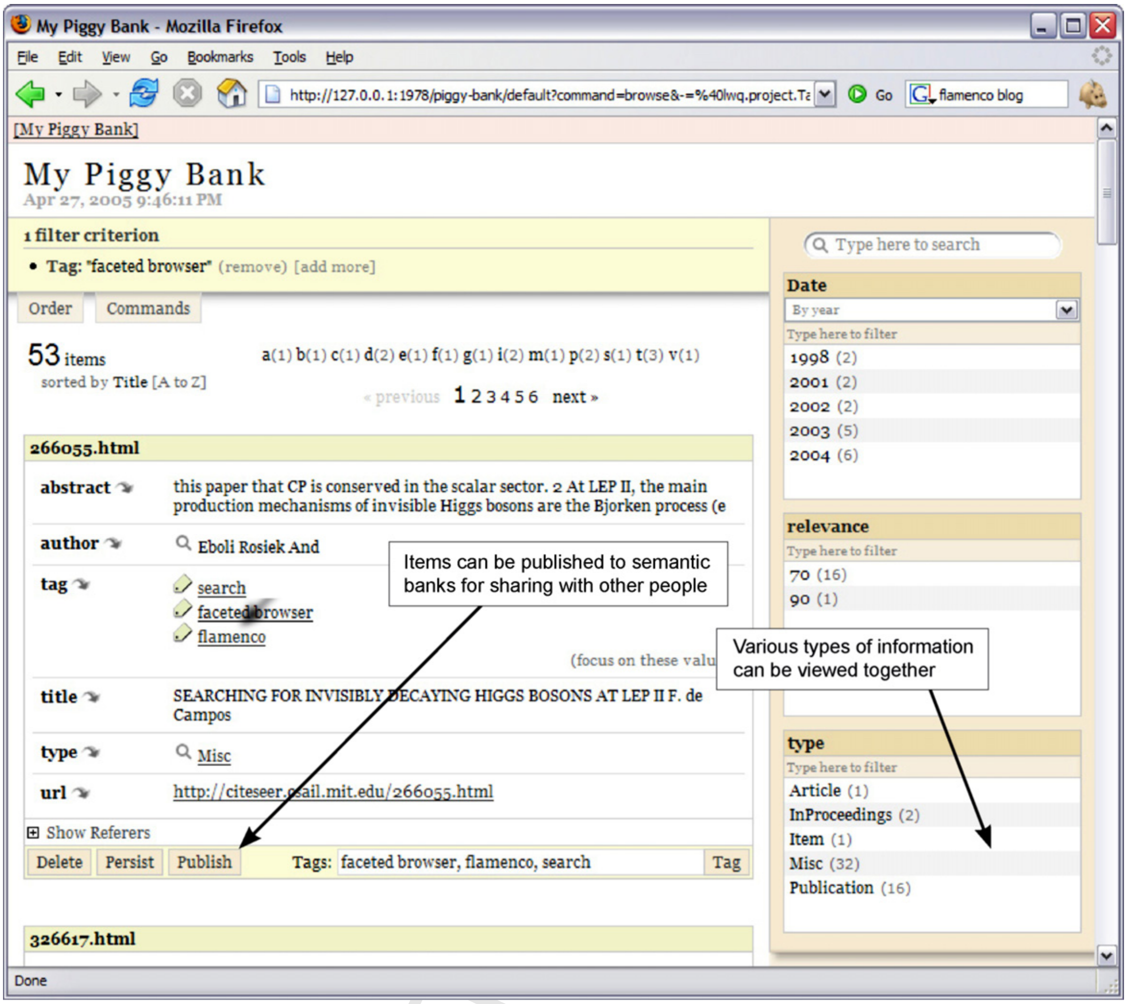


Fig. 5. All locally saved information can be browsed together regardless of each item's type and original source. Items can be published to Semantic Banks for sharing with other people.

with. Furthermore, *since the data she collected is saved in RDF, Alice accumulates Semantic Web information simply by using a tool that improves her use of Web information in her everyday's work*.

2.2. Sharing information

Alice does not work alone and her literature search is of value to her colleagues as well. Alice has registered for an account with the her research group's *Semantic Bank,* which hosts data published by her colleagues.[1] With one click on the "Publish" button for each item, Alice publishes information to the Semantic Bank. She can also publish the several items she is currently seeing using the "Publish All" button. She simply publishes the information in pure form without having to author any presentation for it.

Alice then directs her web browser to the Semantic Bank and browses the information on it much like she browses her Piggy Bank, i.e., by tags, by types, by any other properties in the information, but also by the contributors of the information. She sifts through the information her colleagues have published, refining to only those items she finds relevant, and then clicks on the "data coin" icon to collect them back into her own Piggy Bank.

Bob, one of Alice's colleagues, later browses the Semantic Bank and finds the items Alice has published. Bob searches for the same topic on his own, tags his findings with the same tags Alice has used, and publishes them to the bank. When Alice returns to the bank, she finds items Bob has published together with her own items as they are tagged the same way. Thus, through Semantic Bank, Alice and Bob can collaborate asynchronously and work independently from each other.

3. Design

Having illustrated the user experience, we now describe the logical design of our system—Piggy Bank and Semantic Bank—as well as their dynamics.

[1] To see a live Semantic Bank, visit http://simile.mit.edu/bank/.

Please cite this article in press as: D. Huynh et al., Piggy Bank: Experience the Semantic Web inside your web browser, Web Semantics: Sci. Services Agents World Wide Web (2007), doi:10.1016/j.websem.2006.12.002

3.1. Collect

Core in Piggy Bank is the idea of collecting structured information from various web pages and web sites, motivated by the need to re-purpose such information on the client side in order to cater to the individual user's needs and preferences. We consider two strategies for collecting structured information: with and without help from the Web content publishers. If the publisher of a web page or web site can be convinced to link the served HTML to the same information in RDF format, then Piggy Bank can just retrieve that RDF. If the publisher cannot be persuaded to serve RDF, then Piggy Bank can employ screenscrapers that attempt to extract and re-structure information encoded in the served HTML.

By addressing both cases, we give Web content publishers a chance to serve RDF data the way they want while still enabling Web content consumers to take matter into their own hands if the content they want is not served in RDF. This solution gives consumers benefits even when there are still few web sites that serve RDF. At the same time, we believe that it might give producers incentive to serve RDF in order to control how their data is received by Piggy Bank users, as well as to offer competitive advantage over other web sites.

In order to achieve a comprehensible presentation of the collected RDF data, we show the data as a collection of "items" rather than as a graph. We consider an item to be any RDF resource annotated with rdf:type statements, together with its property values. This notion of an item also helps explain how much of the RDF data is concerned when the user performs an operation on an item.

3.2. Save

Information items retrieved from each source are stored in a temporary database that is garbage-collected if not used for some time and reconstructed when needed. When the user saves a retrieved item, we copy it from the temporary database that contains it to the permanent "My Piggy Bank" database.

In a possible alternative implementation, retrieved items are automatically saved into the permanent database, but only those explicitly "saved" are flagged. This implementation is space-intensive. As yet another alternative, saving only "bookmarks" the retrieved items, and their data is re-retrieved whenever needed. This second alternative is time-intensive, and although this approach means "saved" items will always be up to date, it also means they can be lost. Our choice of implementation strikes a balance.

3.3. Organize

Piggy Bank allows the user to tag each information item with several keywords, thereby fitting it simultaneously into several organizational schemes. For example, a photograph can be tagged both as "sepia" and "portrait", as it fits into both the "effect" organizational scheme (among "black & white," "vivid," etc.) and the "topic" scheme (among "landscape," "still life," etc.). Tagging has been explored previously as an alternative to folder hierarchies, which incur an overhead in creation and maintenance as well as disallow the co-existence of several organizational schemes on the same data [37,38,42].

We support tagging through typing with dropdown completion suggestions. We expect that such interaction is lightweight enough to induce the use of the feature. As we will discuss further in a later section, we model tags as RDF resources named by URIs with keyword labels. Our support for tagging is the first step toward full-fledged user-friendly RDF editing.

3.4. View

Having extracted "pure" information from presentation, Piggy Bank must put presentation back on the information before presenting it to the user. As we aim to let users collect any kind of information they deem useful, we cannot know ahead of time which domains and ontologies the collected information will be in. In the absence of that knowledge, we render each information item generically as a table of property/values pairs. However, we envision improvements to Piggy Bank that let users incorporate on-demand templates for viewing the retrieved information items.

3.5. Browse/search

In the absence of knowledge about the domains of the collected information, it is also hard to provide browsing support over that information, especially when it is heterogeneous, containing information in several ontologies. As these information items are faceted in nature—having several facets (properties) by which they can be perceived—we offer a faceted browsing interface (e.g., [41,43]) by which the user can refine a collection items down to a desired subset. Fig. 5 shows three facets—date, relevance, and type—by which the 53 items can be refined further.

Regardless of which conceptual model we offer users to browse and find the items they want, we still keep the Web's navigation paradigm, serving information in pages named by URLs. Users can bookmark the pages served by Piggy Bank just like they can any web page. They can use the Back and Forward buttons of their web browsers to traverse their navigation histories, just like they can while browsing the Web.

Note that we have only criticized the packaging of information into web pages and web sites in the cases where the user does not have control over that packaging process. Using Piggy Bank, the user can save information locally in RDF, and in doing so, has gained much more say in how that information is packaged up for browsing. It is true that the user is possibly constrained by Piggy Bank's user interface, but Piggy Bank is one single piece of software on the user's local machine, which can be updated, improved, configured, and personalized. On the other hand, it is much harder to have any say on how information from several web sites is packaged up for browsing by each site.

Please cite this article in press as: D. Huynh et al., Piggy Bank: Experience the Semantic Web inside your web browser, Web Semantics: Sci. Services Agents World Wide Web (2007), doi:10.1016/j.websem.2006.12.002

3.6. Share

Having let users collect Web information in Semantic Web form and save it for themselves, we next consider how to enable them to share that information with one another. We again apply our philosophy of lightweight interactions in this matter. When the user explicitly publishes an item, its properties (the RDF subgraph starting at that item and stopping at non-bnodes) are sent to the Semantic Banks that the user has subscribed to. The user does not have fine-grained control over which RDF statements get sent (but the items being handled are already of possibly much finer granularity compared to full webpages). This design choice sacrifices fine-grained control in order to support publishing with only a single-click. Thus, we make our tools appealing to the "lazy altruists", those who are willing to help out others if it means little or no cost to themselves.

Items published by members of a Semantic Bank get mixed together, but each item is marked with those who have contributed it. This bit of provenance information allows information items to be faceted by their contributors. It also helps other members trace back to the contributor(s) of each item, perhaps to request for more information. In the future, it can be used to filter information for only items that come from trusted contributors.

3.7. Collaborate

When an item is published to a Semantic Bank, tags assigned to it are carried along. As a consequence, the bank's members pool together not only the information items they have collected but also their organization schemes applied on those items.

The technique of pooling together keywords has recently gained popularity through services such as del.icio.us [6], Flickr [25], and CiteULike [4] as a means for a community to collaboratively build over time a taxonomy for the data they share. This strategy avoids the upfront cost for agreeing upon a taxonomy when, perhaps, the nature of the information to be collected and its use are not yet known. It allows the taxonomy to emerge and change dynamically as the information is accumulated. The products of this strategy have been termed *folk taxonomies*, or *folksonomies*.

Another beneficial feature of this strategy is that the collaborative effect may not be intentional, but rather accidental. A user might use keywords for his/her own organization purpose, or to help his/her friends find the information s/he shares. Nevertheless, his/her keywords automatically help bring out the patterns on the entire data pool. Our one-click support for publishing also enables this sort of folksonomy construction, intentional or accidental, through Piggy Bank users' wishes to share data.

While a taxonomy captures names of things, an ontology captures concepts and relationships. We would like to explore the use of RDF to grow not just folksoraomfes, but also *folksologies* (*folk ontologies*). For this purpose, we model tags not as text keywords, but as RDF resources named by URIs with keywords as their labels, so that it is possible to annotate them. For example, one might tag a number of dessert recipes with "durian"$_{\text{tag}}$ then tag the "durian"$_{\text{tag}}$ itself with "fruit". Likewise, the user might tag several vacation trip offers as "South-East Asia"$_{\text{tag}}$ and then tag "South-East Asia"$_{\text{tag}}$ with "location"$_{\text{tag}}$. It is now possible to create a relationship between "fruit"$_{\text{tag}}$ and "location" to say that things tagged as "fruit"$_{\text{tag}}$ "can be found at"$_{\text{rel}}$, things tagged with "location"$_{\text{tag}}$. (Arbitrary relationship authoring is not yet supported in Piggy Bank's user interface).

By modelling tags not as text keywords but as RDF resources, we also improve on the ways *folksonomies* can be grown. In existing implementations of text keyword-based tagging, if two users use the same keyword, the items they tag are "collapsed" under the same branch of the taxonomy. This behavior is undesirable when the two users actually meant different things by the same keyword (e.g., "apple" the fruit and "apple" the computer company). Conversely, if two users use two different keywords to mean the same thing, the items they tag are not "collapsed" and hence fall under different branches of the taxonomy (e.g., "big apple" and "new york"). These two cases illustrate the limitation in the use of syntactic collision for grouping tagged items. By modeling tags as RDF resources with keyword labels, we add a layer of indirection that removes this limitation. It is now possible to separate two tags sharing the same keyword label by adding annotations between them, to say that one tag is OWL:differentFrom another tag. Similarly, an OWL:sameAs predicate can be added between two tags with different labels.

In Piggy Bank and Semantic Bank, when two different tags with the same label are encountered, the user interface "collapse" their items together by default. Though the user interface currently behaves just like a text keyword-based implementation, the data model allows for improvements to be made once we know how to offer these powerful capabilities in a user-friendly way.

Of course, folksologies cannot replace formal ontologies. The role of folksologies is to serve as a low-cost starting point from which ontologies can be formalized. Folksologies provide benefits to their communities at every point along the way until the communities need and know how to formalize their ontologies.

3.8. Extend

We support easy and safe installation of scrapers through the use of RDF. A scraper can be described in RDF just like any other piece of information. To install a scraper in Piggy Bank, the user only needs to save its metadata into his/her Piggy Bank, just like she would any other information item, and then "activates" it (Fig. 6). In activation, Piggy Bank adds an assertion to the scraper's metadata, saying that it is "trusted" to be used by the system. (This kind of assertion is always removed from data collected from websites, so that saving a scraper does not inadvertently make it "trusted".)

4. Implementation

In this section, we discuss briefly the implementation of our software, keeping in mind the logical design we needed to support as discussed in the previous section.

Please cite this article in press as: D. Huynh et al., Piggy Bank: Experience the Semantic Web inside your web browser, Web Semantics: Sci. Services Agents World Wide Web (2007), doi:10.1016/j.websem.2006.12.002

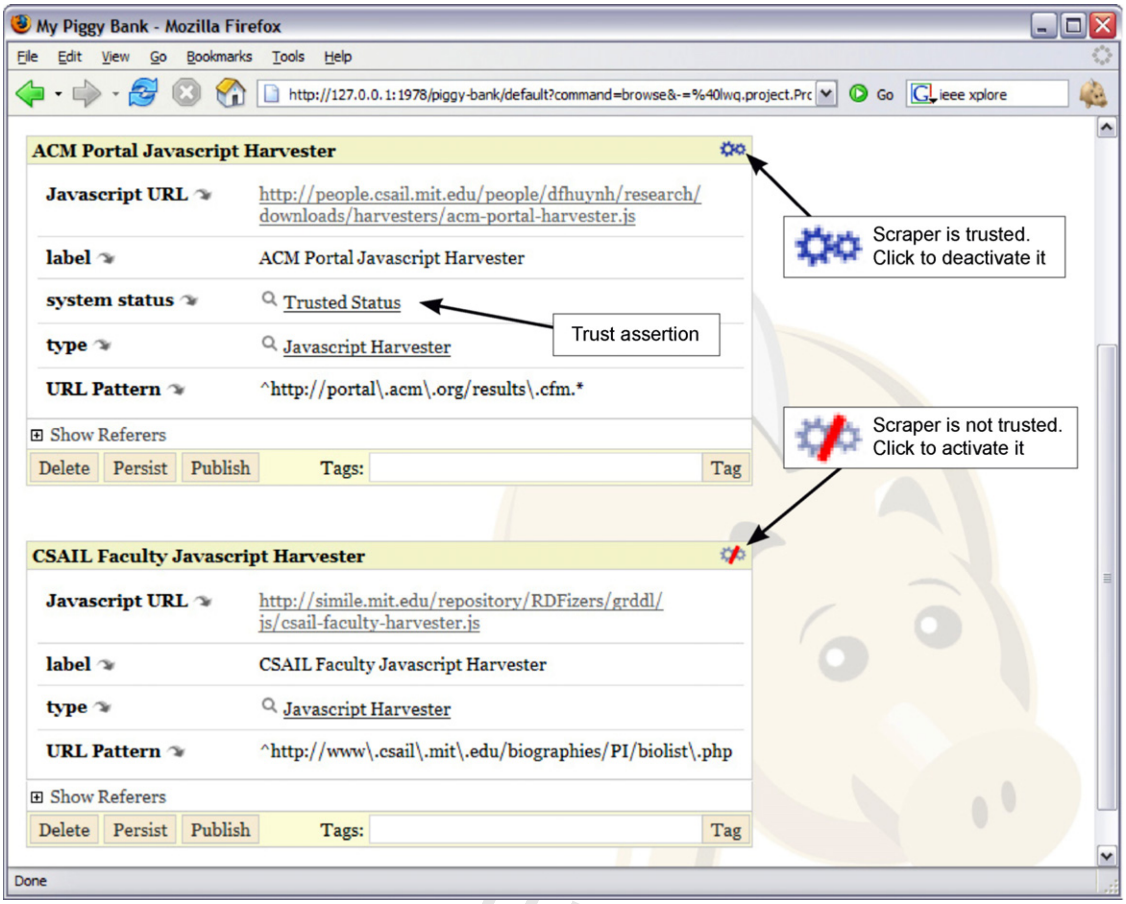


Fig. 6. Installation of a scraper involves saving its metadata and then activating it to indicate that it is trusted to be used within the system.

4.1. Piggy Bank

First, since a core requirement for Piggy Bank is seamless integration with the web browser, we chose to implement Piggy Bank as an extension to the web browser rather than as a stand-alone application (cf. Haystack [39]). This choice trades rich user interface interactions available in desktop-based applications for lightweight interactions available within the web browser. This tradeoff lets users experience the benefits of Semantic Web technologies without much cost.

Second, to leverage the many Java-based RDF access and storage libraries in existence, we chose to implement Piggy Bank inside the Firefox browser [7], as we had found a way to integrate these Java-based RDF libraries into Firefox. By selecting Java as Piggy Bank's core implementation language, we also opened ourselves up to a plethora of other Java libraries for other functionalities, such as for parsing RSS feeds [21] (using Informa [11]) and for indexing the textual content of the information items (using Lucene [3]).

In order to make the act of collecting information items as lightweight as possible, first, we make use of a status-bar icon to indicate that a web page is scrapable, and second, we support collecting through a single-click on that same icon. Piggy Bank uses any combination of the following three methods for collection:

- Links from the current web page to Web resources in RDF/XML [19], N3 [18], or RSS [21] formats are retrieved and their targets parsed into RDF.
- Available and applicable XSL transformations [31] are applied on the current web page's DOM [24].
- Available and applicable Javascript code is run on the current web page's DOM, retrieving other web pages to process if necessary.

Once the user clicks on the data coin icon, we need to present the collected information items to him/her. As mentioned above, we wanted to keep the Web's navigation paradigm by allowing the user to browse collected information as web pages named by URLs. This design choice required Piggy Bank to generate its user interface as DHTML [34]. Since Piggy Bank must generate its DHTML-based user interface *on-the-fly* based on data dynamically collected and saved, we decided to make use of a servlet capable of generating DHTML.[2]

This design turns Piggy Bank into a 3-tier Java-based web server application, containing an RDF database backend, a tem-

[2] The DHTML-based faceted browsing engine of Piggy Bank is Longwell version 2.0. Longwell 1.0 was written by Mark Butler and the Simile team.

Please cite this article in press as: D. Huynh et al., Piggy Bank: Experience the Semantic Web inside your web browser, Web Semantics: Sci. Services Agents World Wide Web (2007), doi:10.1016/j.websem.2006.12.002

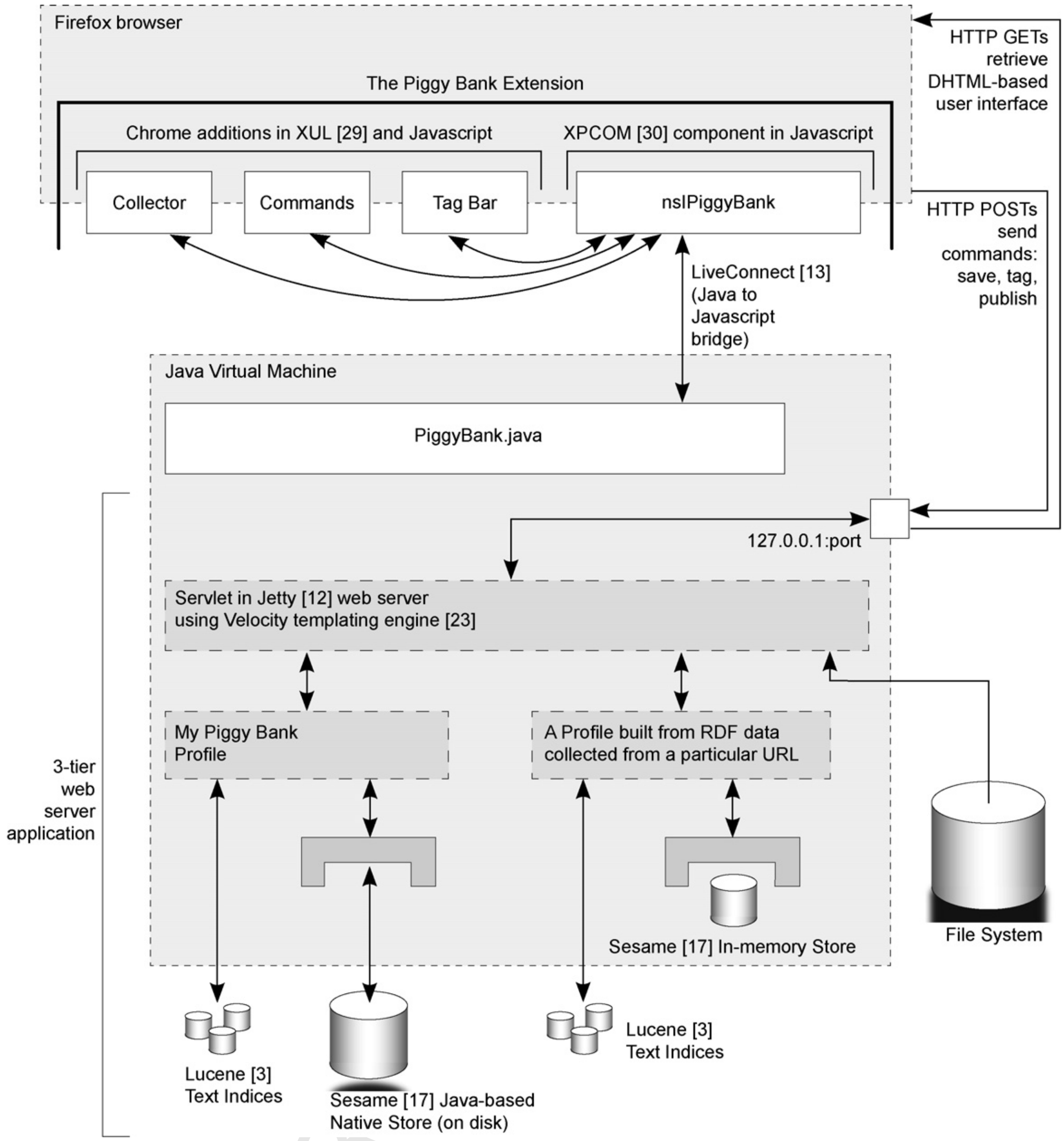


Fig. 7. Piggy Bank's architecture—a web server within the web browser. The embedded Java-based web server resolves queries, fetches data from several backend databases, and generates a DHTML [34]-based user interface on-the-fly using a templating engine. It also processes HTTP POSTs to respond to Save, Tag, and Publish commands. Chrome additions to the Firefox browser detect document loading events, invoke scrapers' Javascript code on document DOMs [24], and provide XUL [29]-based UIs for interacting with the extension. An XPCOM [30] component called nsIPiggyBank written in Javascript provides a bridge over to Piggy Bank's Java code.

plating engine, and a DHTML frontend, all embedded within the Firefox web browser (Fig. 7).

In fact, Piggy Bank has several databases: a permanent "My Piggy Bank" database for storing saved information and several temporary in-memory databases, each created to hold information collected from a different source. The Save command causes data to be copied from a temporary database to the permanent database. Commands such as Save, Tag, and Publish are implemented as HTTP POSTs, sent from the generated DHTML-based user interface back to the embedded web server. Tag completion suggestions are supported in the same manner.

We plan to make the My Piggy Bank database accessible to other extensions in Firefox as well as to other applications so that even more values can be put into and derived from the collected data.

4.2. Semantic Bank

Semantic Bank shares a very similar architecture to the Java part of Piggy Bank. They both make use of the same servlet that serves their DHTML-based faceted browsing user interface. They make use of several profiles for segregating data models. Semantic Bank gives each of its subscribed members a different profile for persisting data while it keeps another profile where "published" information from all members gets mixed together.

Semantic Bank listens to HTTP POSTs sent by a user's piggy bank to upload his/her data. All of the uploaded data goes into that user's profile on the Semantic Bank, and those items marked as public are copied to the common profile. Each published item is also linked to one or more members of the Semantic Bank who have contributed that item.

5. Related work

We will now take a trip back in history to the birth of the World Wide Web, and witness that even at that time, ad hoc solutions were already suggested to combat the highly flexible but still constraining information model of the Web.

5.1. Consumption

When the number of web sites started to accumulate, directories of web sites (e.g., Yahoo! [32]) were compiled to give an overview of the Web. When the number of sites continued to grow, search engines were invented to offer a way to query over all sites simultaneously, substantially reducing concerns about the physical location of information, thereby letting users experience the Web as a whole rather than as loosely connected parts. Capable of liberating web pages from within web sites, search engines still cannot liberate individual information items (e.g., a single phone number) from within their containing pages. Furthermore, because these third-party search engines do not have direct access into databases embedded within web sites, they cannot support structured queries based on the schemas in these databases but must resolve to index the data already rendered into HTML by the web sites.

Another invention in the early days of the Web was web portals which provided personalizable homepages (e.g., My Netscape [14]). A user of a portal would choose which kinds of information to go on his/her portal homepage, and in doing so, aggregate information in his/her own taste. Such an aggregation is a one-time costly effort that generates only one dynamic view of information, while aggregation through Piggy Bank happens by lightweight interactions, generating many dynamic views of information through the act of browsing. During the evolution of the web portal, the need for keeping aggregated news articles up-to-date led to the invention of RSS (originally Rich Site Summary) [21] that could encode the content of a web site chronologically, facilitating the aggregation of parts of different sites by date. RSS was the first effort to further reduce the granularity of the information consumption on the web that achieved widespread adoption. RSS feeds are now used by web sites to publish streams of chronologically ordered information for users do consume. RSS was also the first example of a pure-content format, firmly separating the concern of data production and data consumption and allowing innovative user interfaces to exist (e.g., [16]).

Also early in the history of the World Wide Web came screenscrapers—client-side programs that extract information from within web pages (e.g., stockquotes, weather forecasts) in order to re-render them in some manners customized to the needs of individual users. The news aggregators (e.g., [8]) juxtaposed fragments ripped out from various news web sites together to make up a customized "front page" for each user according to his/her news taste. More recently, client-side tools such as Greasemonkey [9] and Chickenfoot [33] let advanced users themselves prescribe manipulations on elements within web pages, so to automate tedious tasks or to customize their Web experience. Additions to web browsers such as Hunter-Gatherer [40] and Net Snippets [15] let users bookmark fragments within web pages, and Annotea [36] supports annotation on such fragments.

Piggy Bank adopts the scraping strategy but at a platform level and also introduces the use of RDF as a common data model wherein results from different scrapers can be mixed, thus allowing for a unified experience over data scraped from different sources by different scrapers. Piggy Bank is capable of storing more than just XPaths [28] pointing to information items as Hunter-Gatherer [40], and it allows users to extract data rather than annotate documents as Annotea [36] does. Piggy Bank does not rely on heuristics to re-structure information as Thresher [35] does, but rather requires people write easily distributable scraping code. It is possible to make use of Thresher [35] as a scraper writing tool.

5.2. Production

On the production side, HTTP [10] natively supports posting of data to a URL, though it leaves the syntax and semantic of that data as well as how the data is used to the web server at that URL. Web sites have been employing this mechanism to support lightweight authoring activities, such as providing registration information, rating a product, filling out an online purchase order, signing guestbooks, and posting short comments.

A more sophisticated form of publishing is Web logs, or blogs. Originally written by tech-savvy authors in text editors (e.g., [1]), blogs have morphed into automated personal content management systems used by tech-unsavvy people mostly as online journals or for organizing short articles chronologically. Using RSS technology, blog posts from several authors can be extracted and re-aggregated to form "planets".

Unlike blog planets, wikis [27] pool content from several authors together by making them collaborate on the editing of shared documents. This form of collaborative, incremental authoring, while strongly criticized for its self-regulating nature and generally very low barrier to entry [5], has been proven incredibly prolific in the creation of content and at the same time very popular. (Wikipedia [26] is visited more often than the New York Times [2].)

The effectiveness of socially scalable solutions is also evident in the more recent social book-marking services (e.g., del.icio.us [6]) where content authoring is extremely lightweight (assigning keywords) but the benefit of such authoring effort is amplified when the information is pooled together, giving rise to overall patterns that no one user's data can show.

Please cite this article in press as: D. Huynh et al., Piggy Bank: Experience the Semantic Web inside your web browser, Web Semantics: Sci. Services Agents World Wide Web (2007), doi:10.1016/j.websem.2006.12.002

6. Conclusion

In adopting Piggy Bank, users immediately gain flexibility in the ways they use existing Web information without ever leaving their familiar web browser. Through the use of Piggy Bank, as they consume Web information, they automatically produce Semantic Web information. Through Semantic Bank, as they publish, the information they have collected merges together smoothly, giving rise to higher-ordered patterns and structures. This, we believe, is how the Semantic Web might emerge from the Web. In this section, we discuss how the rest of the story might go.

6.1. Scraping the web

Our story is about empowering Web users, giving them control over the information that they encounter. Even in the cases where the web sites do not publish Semantic Web information directly, users can still extract the data using scrapers. By releasing a platform on which scrapers can be easily installed and used, and they can contribute their results to a common data model, we have introduced a means for users to integrate information from multiple sources on the Web at their own choosing.

In this new “scraping ecosystem,” there are the end-users who want to extract Semantic Web information, scraper writers who know how to do so, and the publishers who want to remain in control of their information. We expect that many scraper writers will turn their creativity and expertise at scraping as many sites as they can so to liberate the information within.

The explosion of scrapers raises a few questions. Will there be a market where scrapers for the same site compete on the quality of data they produce? Will there be an explosion of several ontologies for describing the same domain? How can a user find the “best” scraper for a site? Which kinds of site will be more susceptible to scraping?

As a possible scenario, a centralized service could host the metadata of scrapers in order to support easy or automatic discovery of scrapers for end-users while allowing scraper writers to coordinate their work. Such a centralized service, however, is a single point of failure and a single target for attack. An alternative is some form of peer-to-peer scraper-sharing network.

6.2. Information wants to be free

Our system goes beyond just collecting Semantic Web information but also enables users to publish the collected information back out to the Web. We expect that the ease with which publishing can be done will encourage people to publish more. This behavior raises a few questions. How can we build our system to encourage observance of copyright laws? How will publishers adapt to this new publishing mechanism? How will copyright laws adapt to the fine-grained nature of the information being redistributed? Is a Semantic Bank responsible for checking for copyright infringement of information published to it? Will scraper writers be held responsible for illegal use of the information their scrapers produce on a massive scale?

In order to remain in control of their information, one might expect publishers to publish Semantic Web information themselves so to eliminate the need for scraping their sites. They might include copyright information into every item they publish and hold Piggy Bank and Semantic Bank responsible for keeping that information intact as the items are moved about.

Perhaps it is in the interest of publishers to publish Semantic Web information not only to retain copyright over their information but also to offer advantages over their competitors. They can claim to publish richer, purer, more standard-compliant, more up-to-date, more coherent, more reliable data that is more usable, more mixable, more trustable. They can offer searching and browsing services directly on their web sites that are more sophisticated than what Piggy Bank can offer. They can even take advantage of this new publishing mechanism to spread their advertisements more easily.

~~Uncited references~~

~~[12,13,17,23].~~

Acknowledgements

This work is conducted by the Simile Project, a collaborative effort between the MIT Libraries, the Haystack group at MIT CSAIL, and the World Wide Web Consortium. We would like to thank Eric Miller, Rob Miller, MacKenzie Smith, Vineet Sinha, the Simile group, the User Interface Design group, and the Haystack group for trying out Piggy Bank and for their valuable feedbacks on this work. Last but not least, we are in debt to Ben Hyde for having infected us with the idea of a “scraping ecosystem.”

References

[1] 9101 – /News. http://www.w3.org/History/19921103-hypertext/hypertextAVWW/News/9201.html.
[2] Alexa Web Search - Top 500. http://www.alexa.com/site/ds/top_sites?ts_mode=lang&lang=en.
[3] Apache Lucene. http://lucene.apache.org/.
[4] CiteULike: A free online service to organise your academic papers, http://www.citeulike.org/.
[5] Criticism of Wikipedia. http://en.wikipedia.org/wiki/Criticism_of_Wikipedia.
[6] del.icio.us. http://del.icio.us/.
[7] Firefox - Rediscover the web. http://www.mozilla.org/products/firefox/.
[8] Google News. http://news.google.com/.
[9] Greasemonkey. http://greasemonkey.mozdev.org/.
[10] HTTP - Hypertext Transfer Protocol Overview. http://www.w3.org/Protocols/.
[11] Informa: RSS Library for Java. http://informa.sourceforge.net/.
[12] Jetty Java HTTP Servlet Server. http://jetty.mortbay.org/jetty/.
[13] LiveConnect Index. http://www.mozilla.org/js/liveconnect/.
[14] My Netscape. http://my.netscape.com/.
[15] Net Snippets. http://www.netsnippets.com/.
[16] NewsMap. http://www.marumushi.com/apps/newsmap/newsmap.cfm.
[17] openRDF.org - home of Sesame. http://www.openrdf.org/.
[18] Primer - Getting into the semantic web and RDF using N3. http://www.w3.org/2000/10/swap/Primer.html.
[19] RDF/XML Syntax Specifications (Revised). http://www.w3.org/TR/rdf-syntax-grammar/.

[20] Resource Description Framework (RDF)/W3C Semantic Web Activity. http://www.w3.org/RDF/.
[21] RSS 2.0 Specifications. http://blogs.law.harvard.edu/tech/rss.
[22] Semantic Web project. http://www.w3.org/2001/sw/.
[23] Velocity. http://jakarta.apache.org/velocity/.
[24] W3C Document Object Model. http://www.w3.org/DOM/.
[25] Welcome to Flickr - Photo Sharing. http://flickr.com/.
[26] Wikipedia. http://www.wikipedia.org/.
[27] Wiki Wiki Web. http://c2.com/cgi/wik?WikiWikiWeb.
[28] XML Path Language (XPath). http://www.w3.org/TR/xpath.
[29] XML User Interface Language (XUL) Project. http://www.mozilla.org/projects/xul/.
[30] XPCOM. http://www.mozilla.org/projects/xpcom/.
[31] XSL Transformations (XSLT). http://www.w3.org/TR/xslt.
[32] Yahoo! http://www.yahoo.com/.
[33] M. Bolin, M. Webber, P. Rha, T. Wilson, R. Miller, Automation and customization of rendered web pages, UIST 2005~~, submitted for publication~~.
[34] D. Goodman, Dynamic HTML: The Definitive Reference., second ed., O'Reilly & Associates, Inc, 2002.
[35] A. Hogue, D. Karger, Thresher: automating the unwrapping of semantic content from the world wide web, in: Proceedings of WWW 2005.
[36] J. Kahan, M. Koivunen, E. Prud'Hommeaux, R. Swick, Annotea: an open RDF infrastructure for shared web annotations, in: Proceedings of WWW 2001.
[37] M. Lansdale, The psychology of personal information management, Appl. Ergonom. 19 (1) (1988) 55–66.
[38] T. Malone, How do people organize their desks? Implications for the design of office information systems, ACM Trans. Office Information Syst. 1 (1) (1983) 99–112.
[39] D. Quan, D. Karger, How to make a semantic web browser, in: Proceedings of WWW 2004.
[40] M.C. Schraefel, Y. Zhu, D. Modjeska, D. Wigdor, S. Zhao, Hunter gatherer: interaction support for the creation and management of within-web-page collections, in: Proceedings of WWW 2002.
[41] V. Sinha, D. Karger, Magnet: supporting navigation in semistructured data environments, in: Proceedings of SIGMOD 2005.
[42] S. Whittaker, C. Sidner, Email overload: exploring personal information management of email, in: Proceedings of SIGCHI1996.
[43] P. Yee, K. Swearingen, K. Li, M. Hearst, Faceted metadata for image search and browsing, in: Proceedings of CHI 2003.

Please cite this article in press as: D. Huynh et al., Piggy Bank: Experience the Semantic Web inside your web browser, Web Semantics: Sci. Services Agents World Wide Web (2007), doi:10.1016/j.websem.2006.12.002