A Review of Piggy Bank and How it Compares to SADle

Darren Lunn
Human Centred Web Lab
School of Computer Science
University of Manchester
UK

The Semantic Web vision is one where information and presentation are separated in order for automated tools and computer agents to find and extract the information that the user wants more accurately. To be able to do this, machines will need to be able to freely access data and information contained within web documents and for these documents to be reasoned over.

Piggy Bank is a tool which attempts to extract pure data from a web page in order to provide users with some of the information retrieval power of the Semantic Web without fully adopting the tools and technologies that will be required for the Semantic Web to come into existence. SADle is a tool that uses the structure of the web page in order to transcode it so that it better suits the accessibility needs of the user. This paper will look at what Piggy bank can do with the data it extracts from a web page and how this compares with SADle.
SADJe

The aim of the SADJe project is two fold. The first is to investigate ways in which the visual rendering of Web pages can be utilised to expose the structural semantics of a Web page. Accurately identifying the structure of Web page elements will allow for robust transcoding solutions that render the content in a variety of formats. The second is to investigate how visually impaired users overcome the difficulties of browsing the Web in order to complete their tasks. Understanding how users cope with accessing information on the Web can be used as a guide to transcoding applications that help reduce the need for coping and enhance the accessibility of the Web. The SADJe web pages may be found at http://hcw.cs.manchester.ac.uk/research/sadie/.

SADJe Reports

This report is in the series of HCW SADJe technical reports. Other reports in this series may be found in our data repository, at http://hcw-eprints.cs.manchester.ac.uk/view/subjects/sadie.html. Reports from other Human Centred Web projects are also available at http://hcw-eprints.cs.manchester.ac.uk/.
1 Introduction

Tim Berners-Lee’s original vision of the World Wide Web was one where the meaning of information would play an important role, in order for machines to process both information and data automatically [1]. This vision has become known as the Semantic Web, where automated agents can perform tasks such as information retrieval and search much more effectively than current tools using current Web technologies. The Semantic Web is not an alternative to the current Web, but an evolution from the Web as it stands today into the Semantic Web that Tim Berners-Lee envisioned 20 years ago.

Piggy Bank takes this view of an evolving web as the basis of its approach to giving users access to Semantic Web features. Piggy Bank is a tool that extends the functionality of a standard Mozilla Firefox Web Browser\(^1\). This extension is so that users can still use and experience existing web pages, yet at the same time gives the user an improved Web experience when they encounter a page that Piggy Bank is able to extract data from.

Piggy Bank uses the RDF descriptions of pages as a basis for collecting and sharing information. Provided that a page has a RDF description associated with it, a user can select areas of information from the page. The user can then tag or annotate the information and save it for use later, either in a personal repository or as a shared resource. These annotations then allow users to alter how their information is viewed and find their information more easily than traditional bookmarking at a later date.

SADIe\(^2\) is also a Mozilla Firefox tool that allows users to experience existing web pages as they were designed, yet gives the user an improved Web experience when they encounter a page that SADIe can transcoded. The transcoding allows users to adapt a web page to better suit the device that they use to access the web page. SADIe is predominantly targeted at visually impaired computer users who access web pages via a screen reader. SADIe uses ontologies to describe the elements that exist on a web page. The ontologies are used to capture the explicit and implicit information. This captured knowledge is then used by SADIe so that the page can be rearranged and adapted to better suit the user’s needs.

The next section will now look in more detail at what Semantic Web features Piggy Bank can provide users before going onto look at the architecture that achieves this.

---

1\url{http://www.mozilla.org/products/firefox/}

2 Structural-semantics for Accessibility and Device Independence
2 Aims of Piggy Bank

The main feature of Piggy Bank is that it allows the user to gather and collect information. To work effectively, Piggy Bank requires that the data is structured in a way that allows it to get access to what Piggy Bank calls “pure data”. Piggy Bank uses RDF\(^3\) to describe the data. A description can be associated with a web page in one of two ways. The first is that there already exists an RDF/XML description of the web page. Usually this manifests itself as an RSS feed, which is a technique used by websites so that users can see the latest updates to the site. For example, the BBC news website supports RSS. By using an RSS enabled application, whenever the BBC adds a new news article to its website, the RSS will be updated and the user is informed of the new news story without going to the website and finding it. Because RSS describes the content of the page, Piggy Bank can use this to get access to the different areas of data on the page.

The second way Piggy Bank obtains structured data is via a screen scraper. A screen scraper is a way of extracting data from a web page by matching the data found in the HTML to predefined patterns. For example, a page may have a contact number as the last area of text on the page. A Screen scraper would parse the HTML, look for the last \(<p>\) tag and assume that the contents of that tag is the contact number. Piggy Bank relies on users who want to use a page with no RSS feed to write these screen scrapers in JavaScript, which it then uses to extract the data from the page in order for the user to manipulate and adapt the page.

With the “pure” data from a page captured, Piggy Bank allows the user to manipulate it. Essentially, Piggy Bank is an organisation tool. Based on the descriptions of the data that has been captured, Piggy Bank allows the user to view the data in different ways. For example, a news RSS feed will have a date, title and time as well as many other fields. Piggy Bank looks for all the different values that a field may have and then uses these values to present the user with a filtering option. For example, a user could filter out news articles so that only yesterday’s news articles are shown.

As well as filtering, Piggy Bank allows information from different sources to be merged onto the same page. To do this, a user would extract the “pure” data from the first page in a Firefox Browser tab. They would then extract the “pure” data from the second page in a separate Firefox Browser tab. The Piggy Bank tool then allows the two pages to be merged, so that all the information is presented on one page in a consistent manner, giving the appearance that it has all be obtained from the same information source.

For RSS feeds, the merger is done manually by the user. However, for screen scraper scripts that have been written by the user, the information merger can be done automatically by incorporating it into the code. For example, a screen scraper has been written that extracts the data from an ACM library search. When the user then views the abstract of a paper, Piggy Bank automatically extracts the title and uses Cite Seer to produce a list of papers that have referenced the paper the user is looking at.

---

\(^3\)Resource Description Framework
Once the information from the various sources has been collected, the user can then organise it to better suit their needs. This is achieved by annotating or “tagging” an information link that Piggy Bank has discovered. Annotations can be whatever the user thinks is appropriate for the information they have found. When the information is tagged, the user can then save the information to a personal repository locally, or they can publish it on a server so that anybody can look at the information.

Piggy Bank offers users a way of obtaining data from a page and then viewing it and/or merging it in a variety of ways. To better illustrate what Piggy Bank can achieve, the next section will look at an example scenario of how a user might use Piggy Bank.
3 Piggy Bank in Action

The previous section looked at what Piggy Bank can achieve. Now we will provide two examples of using Piggy Bank. The first scenario will demonstrate how Piggy Bank can use the RDF/XML RSS feed of a web site to extract and manipulate data. The second will look at what can be achieved when a specific screen scraper is written from Piggy Bank.

3.1 RSS Feed Scenario

A user is interested in Doctor Who and wants to create a collection of Doctor Who News stories. The user knows of several sites that may provide Doctor Who news. The first is the BBC’s entertainment news website. The user goes to the website and sees that there is an RSS feed for the page and that Piggy Bank can exploit the RSS, as indicated by “pure” data button that appears next to the RSS feed indicator [Figure 1]. The user clicks the “pure” data button and Piggy Bank uses the RSS feed to present the user with a list of all the news stories that were found on the web page. Piggy Bank also displays a way of filtering the information based on fields that were found in the RSS feed and any values that those fields contain [Figure 2].

The user has heard from a fellow Dr Who fan that there was a news story on the 12th November about Dr Who, so the user selects 12/11/05 from the “Date” Filter. This reduces the number of news stories from 16 to 3 and the Dr Who story is the top one [Figure 3]. With the article found, the user can then annotate the article so that they can catalogue and retrieve it easily next time. The annotations also put the articles into a taxonomy so that related information is grouped together. The annotation can be anything that the user likes. In this case, the user decides that the article is about “Dr Who”, “Billy Piper”, “Rose Tyler” and “Sci-fi”. They enter the annotations [Figure 4] and click the tag button [Figure 5]. The user then has the option of saving the information to their own personal repository or publishing it on an online central repository. The user then repeats this process with other entertainment news sites that have an RSS feed associated with them.

At a later date, the user wishes to review all their information about Dr Who. They click on the “My Piggy Bank” link at the top of the Piggy Bank page and are taken to a screen which shows a list of all the tags the user has added to their taxonomy [Figure 6]. The user finds “Dr Who” and selects that to bring up their collection of Dr Who news items. The same user interface is created as before, except this time the information links come from several different sources. As with the previous Piggy Bank screens, the user can filter the information in order to get what they are looking for. In [Figure 7] we can see that the user can filter the articles by the web site the articles came from, in this case the BBC or Dark Horizons, but other filtering techniques are possible.

4http://news.bbc.co.uk/1/hi/entertainment/tv_and_radio/default.stm
3.2 Screen Scraper Scenario

A research student uses the ACM Digital Library\(^5\) several times a day to find papers of interest in their subject area. However, due to a limited amount of time and the vast number of papers available for their field, the student also uses Cite Seer\(^6\) to see how many times the paper has been referenced in other works. If a paper has been referenced a large number of times, then it is probably a paper that is good and therefore worthwhile reading.

Fortunately Piggy Bank has a screen scraper that will combine this effort into an automatic task. Initially the student goes to the Piggy Bank web site\(^7\) and finds the list of available screen scrapers that have been published [Figure 8]. In the list is a screen scraper for the ACM portal. To activate it, the student clicks the activation button and allows Piggy Bank to download the executable code. With the screen scraper installed, the user can then use the ACM digital library as they usually would. That is, they can search for authors and paper titles and view the results that the ACM search engine finds [Figure 9].

When the user finds a paper that they think might be interesting, they click on a link to display the abstract of the paper as per usual. Because a screen scraper has been written for this page, the “pure” data icon appears at the bottom of the browser window [Figure 10]. If the student thinks that the paper may be useful, they can click on the “pure” data button and Piggy Bank will scrape the screen to get the details about the paper. Piggy Bank will then call Cite Seer and obtain a list of all the papers that have cited the initial paper found in the ACM Digital Library. The student can then filter, annotate and save the paper references in the same way as the Dr Who fan could in the example given above [Figure 11].

\(^5\)http://portal.acm.org/dl.cfm
\(^6\)http://citeseer.ist.psu.edu/
\(^7\)http://simile.mit.edu/piggy-bank/
4 A Brief Overview of SADIe

Use of the World Wide Web has grown exponentially over the past two decades and has become an essential part of everyday life for most people. Web pages on the web tend to be designed with data presentation in mind by designers who are trained in traditional print media such as newspapers and magazines, rather than electronic media such as the Web. While these visually attractive sites have helped increase the popularity of the web and can be accessed by a large number of users, they are not suitable for all users. In particular, people with visual impairments have trouble accessing the information that is contained within the page. This is due to web pages being visual-centric, where information is only made available implicitly from how data looks and where it is located.

SADIe is a Mozilla Firefox toolbar extension that attempts to make visual-centric web pages more accessible to visually-impaired users, yet at the same time, allows sighted users to see web pages as they were intended by the designers. This is achieved by making implicit information explicit through the use of a web page’s CSS and XHTML to create an ontology.

The ontology can be created because XHTML tags can contain attributes, such as ID or Class. In web design, these attributes are used with CSS to determine how to display the elements of the page. While designers use these ID and Class values for determining the layout of the page, they are also inadvertently capturing the semantics of the web page. This allows SADIe to build up a semantic representation of the document and its structure.

SADIe uses the ontology of a web page to transcode it in three ways. Defluff is used to remove unnecessary elements from the page, such as images or advertisements. Reorder rearranges the page so that the most important information is near the top of the page where it is easier to access. Finally, Toggle puts the menu at the top of the page where the user can find it easily.

SADIe is implemented as a three-stage process. There is an ontology creation stage, where the semantics of a web page are captured from the XHTML and CSS and properties added to the elements. This is followed by a reasoning stage, where SADIe asks questions about the different kinds of elements that are on the web page, and finally the transcoding stage, where the web page is made more accessible for the user. For a more detailed description of SADIe and how it works, see [5, 2]
5 Comparisons of SADle and Piggy Bank

On first inspection, it appears that both SADle and Piggy Bank are similar technologies. Both SADle and Piggy Bank attempt to capture the data and information from a web page and allow the user to manipulate the data in order to better suit their needs. However, by looking more closely at the goals and architectures of both systems, it becomes apparent that they are not so closely connected.

The main goal of Piggy Bank is for users to collect and share information. A user has to find the information that they want and then use Piggy Bank to annotate, group and merge data in order to better match the users view of what the information is concerned about. A user can place the information they find in a taxonomy to make it easier to retrieve at a later date. This also helps group things together, so a user can access lots of information from several sources that are connected somehow.

The main goal of SADle is for users to access information. SADle assumes that the user is currently viewing a page they think they want to read but cannot access. This may be because the device used to access the page, such as a screen reader, does not have the ability to access the information contained on the page. SADle then transcodes the page to better suit the access device. SADle is not a tool for collecting or sharing information. It is a tool which allows existing information to be accessed by a wider range of users.

As well as the goals of the tools being different, the architectures used to achieve those goals are also quite distinct. Piggy Bank uses screen scraping and RSS to get access to the data that is on the page. The data that is extracted is then matched to a template that described the data that should be found on the page. The data is then displayed in a format that allows it to be annotated, tagged and stored in a taxonomy. The data values obtained are also used for filtering and ordering of the data, to aid with the user’s search.

SADle, on the other hand, tries to capture the structure of the page by creating an ontology. SADle is not interested in data values per se, but what the data actually is and what information it is trying to convey. This different world view of “pure” data for Piggy Bank and structured data for SADle is, I believe, the main difference between the two systems. SADle will look at an element on a page and transcode the element differently depending on what type of element it is. For example menu elements are moved to the top of the page and removable elements are removed from the page completely. Piggy Bank however, only sees data and only uses that data as a means of applying different filtering values. Whether a value is a date, author or title is irrelevant for Piggy Bank. it just extracts the values for the various fields it finds from the screen scraper or RSS feed and allows the user to remove information that does not contain an instance of a particular value.

Whilst screen scraping can give Piggy Bank quick and easy access to the data of a page, its main flaw is that it is dependent upon the layout of the page not changing. For example, in Section 3, one of the Piggy Bank scenarios looked at was using the ACM screen scraper. The screen scraper works because all ACM Digital Library pages follow the same layout. For example, the title of the paper is in the third row of the first table. It would be relatively simple to construct a JavaScript that then traversed the DOM of the page, found the third row of the first table and
extracted the “title”. However, if ACM modified the layout of the page and decided that the publication date should appear in the third row of the first table, the screen scraper would break. It would find some data that it would assume is the title, but in fact it would be the wrong data and would be meaningless to the user.

SADIe uses on ontologies to represent the structure of the page. This takes longer to build than a screen scraper, but by capturing the structure of the page, SADIe is not susceptible to the problem of layout modifications. The title of a paper will always be the title, regardless of its on-screen positioning. ACM could modify their layout as often as they liked and provided they do not change the structure of the page, SADIe will still function correctly.
6 Conclusion

On first usage, Piggy Bank is impressive and gives the user a feeling of using Semantic technologies to improve how they find and use information on the web. The ability to combine data from different sites has the potential to be very useful and the annotation and saving of found links offers a much more powerful and flexible solution to the current bookmark system used in web browsers. The annotation has no constraints and a user can use any words they choose as a tag value. This can be beneficial because users can store information in any way they choose. However, the lack of control and structure of the taxonomy does not promote sharing and consistent collaboration. Some people may annotate articles about the UK as “Britain”, “Great Britain”, “UK”, ‘The UK”, “The United Kingdom” or “The United Kingdom of Great Britain and Northern Ireland”. Essentially they all have the same meaning, so for better sharing of information, some kind of controlled annotation language may be necessary.

Piggy Bank is a useful bookmarking and annotation tool and is a welcome addition to Mozilla Firefox. It provides the user with a very powerful storage and retrieval facility for related information that comes from different sources. However, it is quite a different tool from SADIe in both its goals and how it achieves those goals. Piggy Bank is an information management system. By scraping the screen to fit data into a predetermined template, Piggy Bank allows the user to capture, store and annotate information links so that they are more useful. SADIe is a transcoding tool which adapts information that the user has already found into a format that better suits the device that the user is accessing the web page with. This is achieved by capturing the structure of the page in an ontology, as opposed to pattern matching data from the screen, making SADIe more robust to future changes to the web page than Piggy Bank.
References


A Piggy Bank Screen Shots

Figure 1: BBC Web Site Showing RSS Feeds and Piggy Bank Support
**Figure 2:** The Results of Extracting “Pure” Data from the BBC Web Site

**Figure 3:** The Results of Filtering “Pure” Data from the BBC Web Site
Figure 4: Annotating Found Information

Figure 5: Saving The Annotated Information
Figure 6: Viewing The User’s Taxonomy

Figure 7: Reviewing Dr Who Articles
Figure 8: Installing a Screen Scraper

Figure 9: Finding Papers in the ACM Digital Library
**Figure 10:** Activating Piggy Bank

**Figure 11:** Filtering the Reference Papers