An Architecture to Discover and Query Decentralized RDF Data

Uldis Bojārs[1] and Alexandre Passant[2] and Frederick Giasson[3] and John Breslin[1]

[1] Digital Enterprise Research Institute, National University of Ireland, Galway
[uldis.bojars, john.breslin]@deri.org
[2] Université Paris IV Sorbonne, Laboratoire LaLICC, Paris, France
alexandre.passant@paris4.sorbonne.fr
[3] Zitgist LLC., Quebec City, Canada
fred@fgiasson.com

Abstract. In this paper we describe a distributed architecture consisting of a combination of scripting tools that interact with each other in order to help to find and query decentralized RDF data. Thanks to this architecture, anyone can participate in the collaborative discovery of Semantic Web documents by simply browsing the web. This system is useful for dynamic discovery of RDF content and can provide a useful source of RDF documents for other Semantic Web applications. Key components of this architecture are the Semantic Radar plugin used for discovery of Semantic Web data, Ping The Semantic Web service for aggregating locations of RDF data, and services using RDF data illustrated here with the example of doap:store.

1 Introduction

As the Semantic Web meme is spreading the number of RDF documents available on the Web is constantly growing and so is the number of tools creating and using this information. People create their FOAF[4] profiles, developers describe open-source projects using DOAP[5], and bloggers and bulletin boards provide data from their sites using SIOC[6] [4]. Semantic Web search engines such as Swoogle [6] and SWSE [10] have been developed to help to find this information, and browsers such as Tabulator [2] or Disco[7] can help to navigate through the Semantic Web.

The amount of documents indexed by Swoogle has reached more than 1.4 million but that is a small amount compared to the size of the Web estimated to be more than 11.5 billion documents [8]; the density of these RDF documents on the Web is still low and finding them can be a hard and laborious task. Moreover, an area which these search engines currently do not address is dynamic content, since information on dynamic websites such as blogs and online community sites is being generated at a fast pace. To adapt to this new trend, a new kind of infrastructure is necessary to enable Semantic Web applications to quickly harvest and use this information.

We present a scripting architecture that aims to solve this problem of finding and querying up-to-date decentralized RDF data, in almost real-time. Our goal is to show how simple scripting applications can work together to provide an extensible architecture for applications browsing and querying Semantic Web documents.

The rest of this paper is organized as follows. Section 2 describes our approach and its key features - decentralized scripting and user participation in discovery of Semantic Web data. Similar to the idea of Web 2.0 where tools and interfaces are

[4] http://foaf-project.org/
[5] http://usefulinc.com/doap/
[6] http://sioc-project.org/
[7] http://sites.wiwiss.fu-berlin.de/suhl/bizer/ng4j/disco/

driven by users the collaborative discovery provides a way to let everyone be part of this architecture. In section 3 we describe the complete architecture of the system consisting of the following components: (1) Semantic Radar, a browser extension used to detect presence of Semantic Web content while browsing the web, (2) Ping The Semantic Web (PTSW), a service for aggregating notifications about recently discovered and updated Semantic Web documents and (3) external services such as doap:store that use these data sources to provide browsing and querying interfaces. Finally, we will present a preliminary evaluation of our approach, introduce related work and conclude the paper by presenting some of the future work.

2 Our approach

A decentralized system. Using the philosophy of Unix programming, we decided to work on a set of small scripting application that each focus on a single job and try to do it as good as possible rather that working on a new centralized architecture. Thus, the system can be seen as a synergy of applications and web services assembled together in a "semantic pipeline": one tool will concentrate on finding documents, another on storing their URIs and providing them to consumers of RDF data, while the third will provide RDF query and browsing interfaces.

User participation. The other characteristic that differentiates this system from services such as Swoogle, is that it strongly involves user participation in the way we discover new Semantic Web documents. No preliminary knowledge of RDF or user effort is required, since users just have to browse the Web to be part of this discovery, as we will see in section 3.2. Thus, real users are involved in this architecture of participation and can help to discover the "invisible" Semantic Web.

Data provider. The centric point of our approach is the way to provide found data to users. Using previous discovery of Semantic Web content, but also pinged by other services such as TalkDigger[8], Ping The Semantic Web maintains a list of RDF documents URIs which is constantly updated, acts as a data provider and can be the entry point of a lot of services, as it provides fresh Semantic Web data.

Browsing and querying services. Using the list of URIs maintained by our data provider, external services that browse or query RDF data can be plugged to this architecture. The first implemented service of this kind is doap:store, a search engine dedicated to DOAP projects, described in detail in section 3.4, while other already existing services as SWSE are also using it.

3 Architecture of the System

Our system involves the following components (see Fig. 1):

- Data sources, i.e. RDF documents spread around the Semantic Web, created by people themselves or by some of the tools they are using as SIOC exporters[9];
- Semantic Radar[10], a Firefox plugin that allows anyone to be part of the discovering of Semantic Web documents by simply browsing the Web, using autodiscovery links to find RDF data from HTML pages;
- Ping The Semantic Web[11] (PTSW), a web service that stores a list of RDF document URLs it receives pings about, mainly via the Semantic Radar but also thanks to other services;

[8] `http://talkdigger.com`
[9] `http://sioc-project.org/exporters`
[10] `http://sioc-project.org/firefox`
[11] `http://pingthesemanticweb.com`

– Browsing and querying services, that use the list of documents URIs stored within PTSW, as doap:store[12] to query and browse DOAP projects or SWSE search engine.

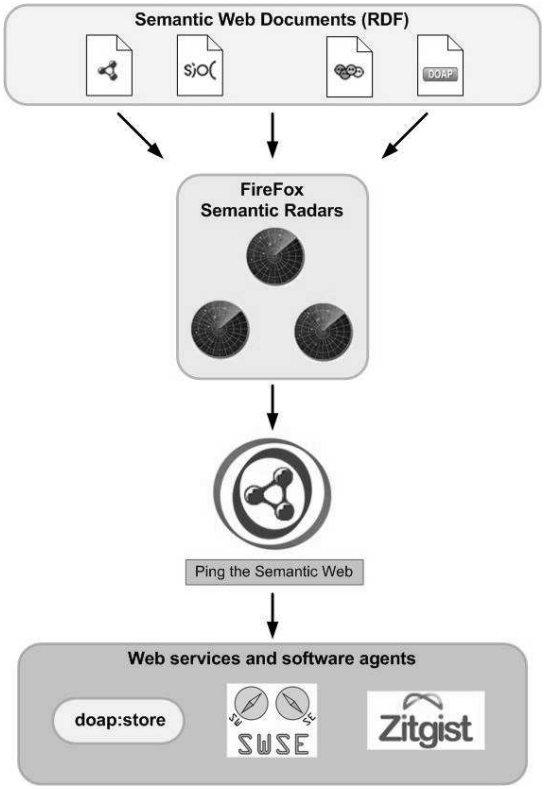

Fig. 1: Global architecture of the system

3.1 Data Sources and RDF Auto-discovery

The starting point of our architecture is RDF data sources that we need to discover. That includes all the RDF information published by people or information systems that they are using. Examples of RDF information on the web include SIOC, FOAF and DOAP data. An important question is how to detect links to RDF information from any web page.

RDF/XML Syntax Specification [1] recommends to use a `<link>` element in in the `<head>` element of HTML pages to point to additional RDF documents instead of directly embedding RDF/XML content in web pages, which may cause validation against DTD to fail, unless using RDFa or other embedded RDF approach. To use this technique the `href` element should point to the URI of RDF/XML content and the `type` attribute should contain the value `"application/rdf+xml"`.

```
<link rel="alternate"
  type="application/rdf+xml" title="RSS 1.0"
  href="http://apassant.net/blog/feed/rdf" />
```

This linking technique is known as RDF auto-discovery and is recommended and used by a number of Semantic Web projects and vocabulary specifications, such as

[12] `http://doapstore.org`

DOAP, FOAF[13], ICRA[14] and SIOC. The same technique with different MIME types is also widely used to point to RSS and Atom feeds. It offers automatic discovery of machine-processable information associated with webpages and applications are aware of that. I.e., web browsers use RSS auto-discovery links to display an RSS icon and to read or subscribe to RSS feeds associated with webpages.

By adding a link to FOAF profile a person enables visitors of the website to discover machine-readable FOAF data using web browser extensions such as Semantic Radar introduced in the next section. Data export tools such as WordPress SIOC plugin[15] use RDF auto-discovery links to facilitate discovery of the information they create.

It is its ease of understanding and implementing that makes RDF auto-discovery a popular choice for indicating presence of RDF data or any other metadata related to a web page.

3.2 Semantic Radar extension for Firefox

Semantic Radar is a Firefox browser extension (written in XUL and JavaScript) which inspects web pages for RDF auto-discovery links and informs a user about presence of them by showing icons in the browser's status bar.

When an auto-discovery link is detected, Semantic Radar examines the `title` attribute of the `link` tag. It uses this attribute as a hint - and only a hint, since it is designed for humans - and if its content matches a pattern associated with one of the data types the application is built to detect, it displays the corresponding icon. Currently supported data types are FOAF, SIOC and DOAP and the patterns used to detect them using the `title` attribute are simply "FOAF", "SIOC" and "DOAP".

This function makes users aware of the invisible Semantic Web that is part of web pages they are exploring every day. An important aspect of the tool is that it is not limited to a particular ontology (there were separate tools for detecting FOAF or other ontologies before that) but provides a generic way to detect various types of documents related to auto-discovery links and can be extended to cover more types of data in the future.

Semantic Radar allows users to click an icon and see this data in a user-friendly RDF browsing interface making it better understandable for human users. As such it can play a role in education and outreach of the Semantic Web to classic Web users and developers. By installing a simple browser plugin users can see what Semantic Web documents are associated with webpages and can explore this information without a need to look at raw RDF data. This is the first motivation for users to install this extension. E.g., when browsing a weblog that links to author's FOAF profile a user can get additional information about the author and his social relations.

The second function of Semantic Radar - "pinging" or sending notifications - is an important part of our architecture for collaborative discovery of Semantic Web data. Whenever a RDF auto-discovery link is discovered by the application it sends a "ping" to PTSW service - described in the next section - which collects and aggregates these notifications. As a result, users are building a "map" of Semantic Web documents, once again without additional effort. This provides a second motivation to install the plugin.

The ping function of Semantic Radar can be switched off at any time by clicking the radar icon in the status bar or via extension preferences thus addressing possible

[13] `http://rdfweb.org/topic/Autodiscovery`
[14] `http://www.icra.org/systemspecification/#specificLink`
[15] `http://sioc-project.org/wordpress`

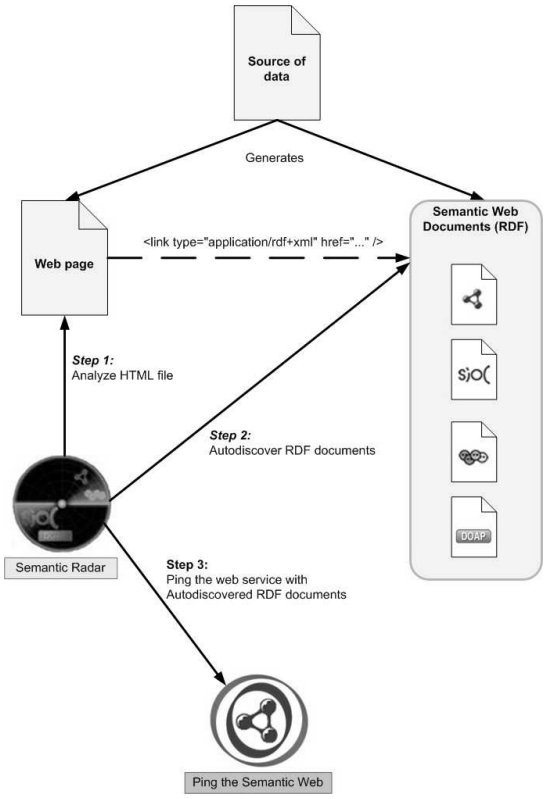

Fig. 2: Overview of Semantic Radar

privacy questions. Additional safe-guards are provided by creating a *"black-list"* of URLs that pings will not be sent about, i.e., when browsing intranet pages.

Joint work of the users browsing the web using Semantic Radar extension and of PTSW service makes collaborative discovery of RDF documents possible. An important characteristic of this process is that the pings are generated by real users. Brin [5] uses an intuitive model of random walkers on the web to explain Google PageRank (the probability of a "random walker" visiting a web page is its PageRank). In collaborative discovery of information there are real "walkers" browsing the information that we can assume they are interested in. Thus the ping summary directly indicate not only presence of pages but also how popular they are.

3.3 Ping The Semantic Web Service

Ping The Semantic Web (PTSW) is a web service that acts as a multiplexer for RDF document notifications. It collects and archives notifications about recently updated or created RDF documents and HTML documents with RDF auto-discovery links, and provides an up-to-date list of Semantic Web documents to services (e.g., web crawlers, software agents) that request it. Similar web services are already widely used for aggregating changes to web feeds, with weblogs.com processing millions of pings each day. PTSW is dedicated to Semantic Web documents and all the sources they may come from: blogs, databases exported in RDF, hand-crafted RDF files, etc. This service can work together with the Semantic Radar or receive pings directly from content provides that let it know about new documents they produce. E.g., TalkDigger and revyu.com send pings for every RDF document they produce to ensure that the information is up-to-date even if nobody browses them using Semantic Radar.

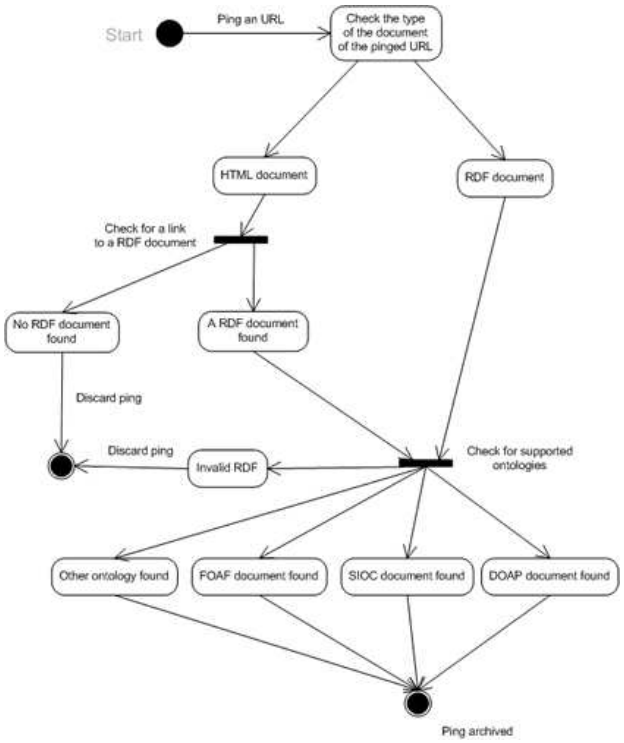

Fig. 3: Pings workflow in PTSW

Applications ping PTSW using one of the ping interfaces provided - REST [7] or XML-RPC. Once the service receives a ping and checks that the document is either RDF or HTML document with RDF auto-discovery links, it parses discovered RDF document(s) to categorize them in one or more of the following 5 categories based on presence of classes and properties from these ontologies: RDFS, OWL, SIOC, FOAF and DOAP. This list can be extended to other vocabularies in the future.

We decided to concentrate on FOAF, SIOC and DOAP first because they are widely available on the Web, and are created by end-users, content management tools and large websites such as FOAF from LiveJournal, SIOC from online community sites and TalkDigger, and DOAP from people and companies describing their software projects.

PTSW provides a live export of its list of URLs which can be used by software agents to find RDF documents. This list can be filtered according to type (vocabularies and ontologies used), date of update, RDF serialization, etc. The categorization of RDF documents by various types is used to help applications to get a list of documents they can understand and are interested in. This way we are minimizing the work of the software that requests a list, so that it can concentrate on manipulating the data instead of searching for it.

PTSW acts as one of the first steps of the Semantic Web food chain: Semantic Web search engines such as SWSE and Swoogle can use data from PTSW in order to quickly update recently changed RDF documents and to find next documents to index, using `rdfs:seeAlso` relationships that can exist from one document to another, and other third-party applications can use it to get an up-to-date list of RDF documents. Currently PTSW uses incoming pings to detect updated RDF documents. The service does not poll RDF documents for updates, but may be extended to do so.

3.4 doap:store

We use doap:store[16], a search engine dedicated to DOAP projects, throughout this paper as an example of a service that uses the list of RDF documents found by the Semantic Radar and PTSW. It is the first service to use PTSW as its main source of information. Since DOAP data are now created and used by many open-source developers and there was no easy way to find a project based on its DOAP description and metadata, we decided that the data collected by PTSW provide a good opportunity to write such a service.

Every hour a Python script gets the list of latest pings received by PTSW and categorised as DOAP documents. It parses the list of URLs, retrieves associated RDF documents and puts them in a local triple store, using the 3store API [9]. Since 3store supports contexts, projects can be easily updated when PTSW receives new pings: content of the old RDF file is removed from the store and replaced by the updated content, so that project descriptions are constantly up-to-date.

Contrary to existing software project directories such as Freshmeat[17], the main distinguishing feature of doap:store is its distributed approach. In regular project directories users have to register at a service and then describe their project using some specific forms, in doap:store they just need to publish a DOAP description of their project on the Web. Then they can ping PTSW directly or have a Semantic Radar enabled Firefox browser ping PTSW when they or visitors browse the webpage.

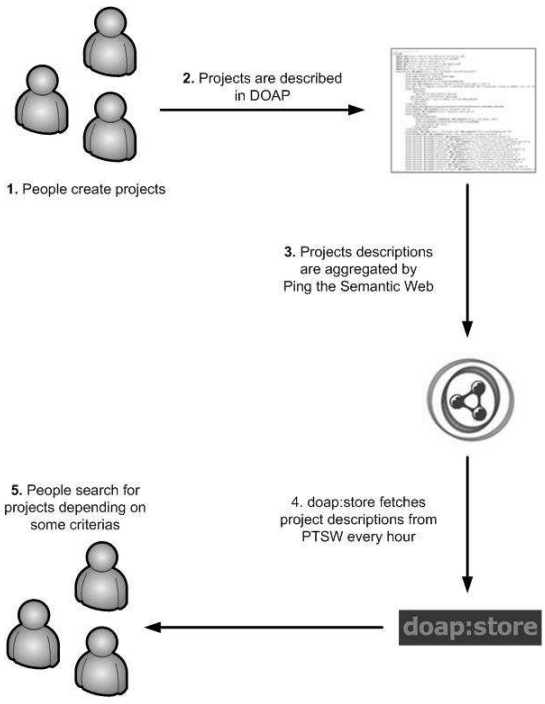

Fig. 4: Information flow in doap:store

We believe that this approach shows what the Semantic Web can offer to both data providers and end-users as:

[16] `http://doapstore.org`
[17] `http://freshmeat.net`

– project maintainers just have to provide a single project description by creating it in a machine-readable format - DOAP - that can be understood by different tools such as doap:store and any generic RDF browser;
– they keep control of the project information since they don't need to publish it on various places where they will not be in control of their data anymore;
– end-users can get an immediate benefit of it, since updated description can be immediately seen in previously mentioned tools.

doap:store is written in PHP and is a quite a small application with about 600 line of code including HTML templates. One of the reasons that helped to realize it so quickly is that it does not have to deal with data discovery since this is managed by previous steps of the application chain introduced in this paper: Semantic Radar and PTSW. Therefore it only needs 10 lines of Python code to integrate and update data in its local database.

The interface offers various ways to query and browse documents:

– using a basic search engine, that allow to retrieve projects by name (`doap:name`), description (`doap:description` and `doap:shortdesc`), both of them, or by hostname (using the URI of the RDF graph corresponding to a `doap:Project`);
– using a tagcloud of programming languages. Tags are retrieved from DOAP files thanks to the `doap:programming-language` property, and then mashed-up to solve case variations and provide a case-insensitive tagcloud;
– using a YubNub[18] command line. Thus, doap:store can be queried with as simple queries as `doap name=rdf`, from web browsers search engines, or other YubNub tools;
– using SPARQL. For most advances users, a SPARQL endpoint is available, offering the way to query the data store or creating new documents from existing content of the data store using SPARQL `CONSTRUCT` instruction.

4 Evaluation

In order to evaluate our system, we made a short comparison of the number of files found by different search engines[19]. Swoogle identifies 416 documents using the DOAP ontology, while a Google search for `doap filetype:rdf` returns 448 documents. PTSW has matched 527 DOAP files, while SPARQL queries addressed to doap:store returns 446 document (using a graph query to find URIs that documents containing at least one instance of `doap:Project`) and 879 projects.

This comparison shows only data which are relevant to doap:store, i.e. a small amount of all RDF data available on the Semantic Web, but one important thing to notice is that this system provides fresh, almost real-time data. To illustrate this, the latest file indexed by Swoogle is from the mid-February 2007, while the latest new file registered in our system was retrieved at the beginning of April 2007. This is even more true with SIOC data since PTSW can record it as soon as data is created - if the blog engine adds it to the list of its pinging services.

PTSW currently has more than 7M documents indexed. A detailed evaluation of different types of data collected by Ping The Semantic Web is outside the scope of this paper and is planned for future work.

5 Related Work

Wiki pages, such as FOAFBulletinBoard[20], were one of the first tools to collect together a number of URIs of RDF documents. This approach works well while

[18] `http://yubnub.org`
[19] On March 30, 2007
[20] `http://rdfweb.org/topic/FOAFBulletinBoard`

there is only a small list of documents with every person adding one or two of them. Its disadvantages are: (1) these lists are usually not maintained (since it requires user intervention) and, as a result, loose quality over time; (2) it is not feasible to manually add URIs when many Semantic Web documents are being created at a fast pace, such as with blog data export in RDF.

SWSE and Swoogle are search engines for the Semantic Web. Swoogle [6] currently contains information on more than 1.4 million RDF documents and is used to find ontologies and Semantic Web documents. SWSE [10] crawls and indexes different types of web documents (XHTML, Atom, etc.), converts them to RDF and provides a user interface to help locate and navigate this information. They may provide APIs to access information indexed but the main use of these services is people using a web interface to query for data.

PiggyBank [11] is a Firefox browser extension which allows to collect RDF data in a distributed fashion. Harvest [3] is an early system that can be used to gather information from diverse repositories to build, search, and replicate indexes, and to cache objects as they are retrieved across the Internet.

Our architecture does not aim to replace Semantic Web search engines and concentrates on one task only - to discover RDF documents on the web and supply applications with a high quality list of Semantic Web documents. Its main use is to retrieve lists of RDF documents in a machine readable form using the API provided. An important difference from existing work is that a large number of users participate in a collaborative discovery of resources on the Web, and that a browser plugin is only the first step of a larger architecture.

6 Conclusion

In this paper, we described an architecture involving user participation in order to discover Semantic Web documents by simply browsing the Web, creating an up-to-date database of RDF documents which can be used by search engines and Semantic web applications to provide search and user-friendly services over this information.

We have shown how a combination of simple scripts can facilitate discovery of distributed Semantic Web data. Benefits of a number of independent applications acting together are that every application is good at a particular task and that new applications can be added to this pipeline because of open data formats used.

End-user applications such as doap:store can use a database of RDF documents provided and build user friendly applications. By bridging the gap between creators of Semantic Web data and the applications that use them, we expect this architecture to provide incentives to create more data and better applications, making the Web of Data into a reality.

While one of the principles of Web 2.0 is that tools and interfaces are driven by users [12], our framework realizes this by providing methods that allow everyone to be part of the Semantic Web initiative. This is a very important difference from blog ping aggregators because collaborative discovery involves and relies upon user participation - the architecture of participation in Web 2.0 terms. Through this architecture, we hope to see how user activity can lead to an enrichment of Semantic Web interfaces to distributed data.

7 Future Work

The future work regarding this architecture has two main aspects.

First, we will try to improve tools and architecture. For example, future versions of Semantic Radar should be able to let users define services other than PTSW they want to send pings to: that way, users could create their own pinging service that will make some specific actions when it receive a new ping. PTSW can also be

improved, possibly including news types of data sources, such as SPARQL endpoints and metadata embedded in webpages. We also hope that other third party services using PTSW will be deployed, as we did with doap:store. Various search engines and front-ends can be created, such as the Zitgist search engine to be released soon.

Performance is not an issue for PTSW now but with a growing number of pings scalability can become a challenge for PTSW. We plan to address it by distributing work across a number of servers as required.

Next, second part will consist of more detailed analysis of data collected. Regarding collaborative discovery, an interesting case study would be to see if there is connections between navigation path and Semantic Web documents relationships. Thus, we could see if people browse HTML documents and follow paths that are related to the Semantic Web metadata they are associated to. To do that we will parse RDF documents registered by PTSW and extract connections between Semantic Web documents. We could also make further research about provenance of namespaces / classes per domain to identify various clusters of metadata.

8 Acknowledgements

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

References

1. D. Beckett. Rdf/xml syntax specification. w3c recommendation, 2004.
2. T. Berners-Lee, Y. Chen, L. Chilton, D. Connolly, R. Dhanaraj, J. Hollenbach, A. Lerer, and D. Sheets. Tabulator: Exploring and analyzing linked data on the semantic web. In *Proceedings of the 3rd International Semantic Web User Interaction Workshop*, 2006.
3. C. M. Bowman, P. B. Danzig, D. R. Hardy, U. Manber, and M. F. Schwartz. The Harvest information discovery and access system. *Computer Networks and ISDN Systems*, 28(1–2):119–125, 1995.
4. J. G. Breslin, A. Harth, U. Bojars, and S. Decker. Towards Semantically-Interlinked Online Communities. In *The 2nd European Semantic Web Conference (ESWC '05), Heraklion, Greece, Proceedings*, May 2005.
5. S. Brin and L. Page. The Anatomy of a Large-Scale Hypertextual Web Search Engine. *Computer Networks*, 30(1-7):107–117, 1998.
6. L. Ding and T. Finin. Characterizing the semantic web on the web. In *5th International Semantic Web Conference*, 2006.
7. R. T. Fielding. Architectural styles and the design of network-based software architectures. *PhD dissertation,Dept. of Computer Science, Univ. of California, Irvine, Calif.*, 2000.
8. A. Gulli and A. Signorini. The indexable web is more than 11.5 billion pages. In *International World Wide Web Conference*, 2005.
9. S. Harris and N. Gibbins. 3store: Efficient bulk rdf storage, 2003.
10. A. Harth, J. Umbrich, and S. Decker. MultiCrawler: A Pipelined Architecture for Crawling and Indexing Semantic Web Data. In *5th International Semantic Web Conference*, 2006.
11. D. Huynh, S. Mazzocchi, and D. Karger. Piggy bank: Experience the semantic web inside your web browser. In *4th International Semantic Web Conference*, pages 413–430, 2005.
12. T. O'Reilly. What is web 2.0: Design patterns and business models for the next generation of software. *O'Reilly Nework*, 2005.