Linkator: Enriching Web Pages by Automatically Adding Dereferenceable Semantic Annotations

Samur Araujo[1], Geert-Jan Houben[1], and Daniel Schwabe[2]

[1] Delft University of Technology, P.O. Box 5031, 2600 GA Delft, The Netherlands
[2] Informatics Department, PUC-Rio Rua Marques de Sao Vicente, 225, Rio de Janeiro, Brazil
{s.f.cardosodearaujo,g.j.p.m.houben}@tudelft.nl,
dschwabe@inf.puc-rio.br

Abstract. In this paper, we introduce Linkator, an application architecture that exploits semantic annotations for automatically adding links to previously generated web pages. Linkator provides a mechanism for dereferencing these semantic annotations with what it calls semantic links. Automatically adding links to web pages improves the users' navigation. It connects the visited page with external sources of information that the user can be interested in, but that were not identified as such during the web page design phase. The process of auto-linking encompasses: finding the terms to be linked and finding the destination of the link. Linkator delegates the first stage to external semantic annotation tools and it concentrates on the process of finding a relevant resource to link to. In this paper, a use case is presented that shows how this mechanism can support knowledge workers in finding publications during their navigation on the web.

Keywords: auto-linking, semantic annotation, semantic link, dereferencing, dynamic links, navigation.

1 Introduction

The links between HTML pages offer the main mechanism for users to navigate on the web. It allows a user, with a simple mouse click, to go from one page to another. However, most websites only contain links that are considered essential for the functioning of the web site, therefore leaving some potentially relevant terms on the web pages unlinked. This can be observed specially in the long tail of the web, where pages are created manually, and where adding extra links can be a laborious and time-consuming task. Because of this situation, a recurring scenario has risen when users browse web pages: they very often select a piece of text (e.g., a telephone, an address, or a name) from a web page and copy and paste it into search engine forms, as the only reasonable procedure available for accessing relevant resources related to those terms. However, this procedure may be significantly reduced, and the navigation improved, by automatically adding links semantically related to those terms.

The problem of automatically adding links to an existing web page can be divided in two main tasks. First, identifying candidate terms (anchors) for adding links – typically they denote concepts in which the user is interested. Second, identifying a

© Springer-Verlag Berlin Heidelberg 2010

web resource to be the link target. The first task can be solved by using information extraction techniques for identifying candidate terms to be linked. Two major sub-problems here are to determine whether a term should be linked, for avoiding "overlinking", and to disambiguate candidate terms to the appropriate concepts, since the terms can have different meanings and consequently demand different links. The second task, that is the focus of this paper, demands using an external source of knowledge in order to discover links related to the terms – actually, the concepts – that were found in the first task. The major sub-problem here is to select a source of data for finding the destination to the link. It can be achieved by querying a pre-defined knowledge base (e.g. Wikipedia) or by querying a distributed and wider data space such as the Semantic Web. The latter approach requires also a pre-selection of the sources to be consulted, since only a few sources in the whole data space can contain relevant knowledge about the concept being exploited.

Although several solutions are being proposed to automatically add links to web pages, they focus on the first part of the problem, i.e. to disambiguate keywords in the text of the page, and typically select the target of the link to be in a single website (e.g., Wikipedia or DBpedia). However, the full problem is only solved after finding the most significant destination for the link. For instance, [3, 4, 5, 6] are focused on linking keywords on web pages to Wikipedia articles, which reduces to disambiguating terms using the Wikipedia as a knowledge base. In spite of their relevance, Wikipedia articles are not always the most adequate objects to link to. These approaches support relatively well users that are interested in, for example, encyclopedic knowledge, however they do not adequately support for example users that are shopping and need to find more information about products, or knowledge workers that are interested in finding bibliographic references for their research.

This paper introduces *Linkator*[1], a framework that uses Semantic Web technology to build dereferenceable semantic annotations, which in this paper are called *semantic links*. It shows how information extraction technology can be composed with semantic technology for automatically and semantically annotating terms in web pages, and subsequently exploiting these semantic annotations to define a target of the link. Linkator expects the extractor component to have the intelligence to disambiguate the terms and annotate them properly, and focuses on determining the appropriate target of the link.

The paper shows the use of the Linked Data cloud[2] as an external source of knowledge for discovering link destinations for the recognized terms. With Linked Data the destination of a link can be semantically determined, instead of restricting it to just one source on the web. The source in the Linked Data to be queried and the query itself are defined based on the semantics of the annotation of the semantic link. By combining these two processes, given an input document, the Linkator system has the ability to identify, on the fly, the important concepts in a text, and then link these concepts to semantically related resources on the web in a contextually relevant way. Later in this paper we will present an example of how Linkator can be used to support knowledge workers.

[1] Linkator prototype is available at:
`http://www.wis.ewi.tudelft.nl/index.php/linkator`

[2] Linked Data - `http://linkeddata.org/`

2 Related Work

2.1 Auto-linking

Augmentation of text with links is not novel. Hughes and Carr [6] discussed the Smart Tag system, a Microsoft agent for automatically enriching documents with semantic actions. Google's AutoLink is another tool for augmenting web pages with links. It uses a pattern-based approach to detect numbers related to entities on web pages, such as street addresses, ISBN numbers, and telephone numbers, and to add links to them. In this tool, users can select an action to be performed after a click action, however it is limited in the number of recognizable entities and the number of sources to be linked to.

Some authors proposed to augment web pages by adding links to Wikipedia articles, therefore having an impartial (e.g., not commercially oriented) target for the link and still adding value to the user navigation. In [3, 4, 5, 6] the same approach for the problem is used. They integrate a term extraction algorithm for automatically identifying the important terms in the input document, and a word sense disambiguation algorithm that assigns each term with the correct link to a Wikipedia article. NNexus [14] proposes an approach that works for general sources of data. Basically, it builds a concept graph of terms extracted from a specific knowledge base and indexes it with the destination document. Then that index is exploited for linking a concept with a document.

Another related research area is that of link recommendation systems. Whereas the idea is similar, these systems are typically oriented towards recommending additional links to entire web pages, as opposed to specific items within a page. Whereas some of the algorithms may be adapted for Linkator's purposes, we have not focused on this aspect here.

2.2 Semantic Annotations

Annotating existing and new documents with semantics is a requirement for the realization of the Semantic Web. A semantic annotation tags an entity on the web with a term defined in an ontology. The annotation process can be done manually or automatically, and the latter demands a specialized system to accomplish the task. The automatic process of annotating is composed basically of finding terms in documents, mapping them against an ontology, and disambiguating common terms. The systems that solve this problem differ in architecture, information extraction tools and methods, initial ontology, amount of manual work required to perform annotation, and performance [7]. The result of the annotation process is a document that is marked-up semantically. For that concern, some markup strategies were proposed. Microformats[3] is an approach to semantic markup for XHTML and HTML documents, that re-uses existing tags to convey metadata. This approach is limited to a few set of published Microformats standards. Moreover, it is not possible to validate Microformats annotations since they do not use a proper grammar for defining it. An

[3] `http://microformats.org/`

evolution of Microformats is eRDF (embedded RDF)[4], an approach for annotating HTML pages using RDF[5], however it faces the same criticism than Microformats, since they use the same strategy for annotating pages. Another approach for semantically annotating pages is *RDFa*[6] (Resource Description Framework - in - attributes). RDFa is a W3C Recommendation that adds a set of attribute level extensions to XHTML for embedding RDF metadata within web documents.

2.3 SPARQL Endpoint Discovery

Querying the Semantic Web implies querying a distributed collection of data. Federated querying over linked data has been addressed in [8, 9, 10, 11, 12, 13]. Among others, a part of this problem is related to selecting the proper sources of data to be queried. For this concern, two approaches stand out. The first approach requires the designer of the query to specify, declaratively, in the body of the query, which sources of data should be queried [9, 11]. In those cases, the endpoints to be queried are fixed and pre-determined by the user. The second approach tries to select the sources automatically, by finding correspondences between the terms mentioned in the query and the sources available in the Semantic Web. For instance, [8, 12] exploit the fact that recursively dereferencing resources mentioned in the query provides the data that then can be used for solving the query. The authors point out that this approach works well in situations where incomplete results are acceptable, since the dereferencing process does not reach all available graphs that match with the query pattern. Also, this approach is limited to a subset of SPARQL queries, since some elements must be present in the query in order to trigger this mechanism.

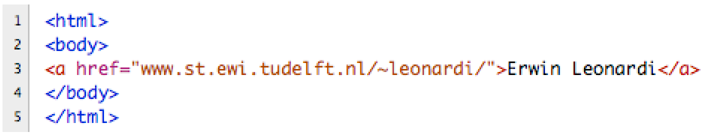
Another approach that tries to exploit such a correspondence uses statistics about the data in the SPARQL endpoints. For instance, [10] summarizes the endpoint data into an index containing statistics about the data space. In order to determine relevant sources, it tries to locate, for each triple pattern in the query, which entry in the index matches it. The selection of the endpoint is determined if the pattern matches with an entry in the index. In [13] middleware is used that catalogs the instances and classes in each endpoint. This middleware selects the right endpoint by matching the resources used in the query with the resources registered in the catalog.

3 Semantic Link – Definition

In an HTML page, an HTML link is denoted by the HTML tag *A*. Normally, it addresses a specific URL which when clicked triggers an HTTP request that retrieves an HTML document that a browser can render in a human-readable representation of this URL. Fig. 1 shows an example of a conventional HTML link. Once clicked, this link redirects the user to the URL described in the *href* attribute: *www.st.ewi.tudelft.nl/~leonardi/*

In these cases, HTML links are defined through an explicit intervention of an author, at the time the page is created or programmed.

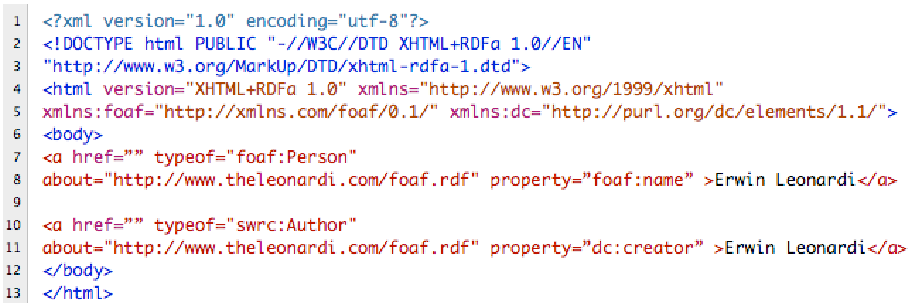
[4] `http://research.talis.com/2005/erdf/wiki/Main/RdfInHtml`
[5] `http://www.w3.org/TR/rdf-primer/`
[6] `http://www.w3.org/TR/xhtml-rdfa-primer/`



```
<html>
<body>
<a href="www.st.ewi.tudelft.nl/~leonardi/">Erwin Leonardi</a>
</body>
</html>
```

Fig. 1. Example of a conventional HTML link

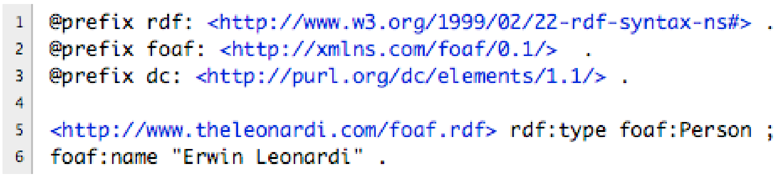
Definition 1: A *semantic link* is an HTML tag *A* that is semantically annotated with *RDFa*, which implies that *RDF* triples are associated to the link. The *semantic link* must contain the attribute *property* or *rel,* which is defined in the RDFa specification; it semantically relates the link to another resource or content. The triples associated to the *semantic link* are determined by the semantics defined in the RDFa specification[7]. Linkator uses the semantics of these triples to compute, dynamically, the URL of the link. Based on the semantics of these triples, it selects sources in the Linked Data cloud to search for a URL for the link. The next example (in Fig. 2) shows how two links with the same anchor (e,g., "Erwin Leonardi") can take the user to distinct pages based on the semantics in the link: Erwin Leonardi's Facebook page and Erwin Leonardi's DBLP[8] publications page.



```
1  <?xml version="1.0" encoding="utf-8"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML+RDFa 1.0//EN"
3  "http://www.w3.org/MarkUp/DTD/xhtml-rdfa-1.dtd">
4  <html version="XHTML+RDFa 1.0" xmlns="http://www.w3.org/1999/xhtml"
5  xmlns:foaf="http://xmlns.com/foaf/0.1/" xmlns:dc="http://purl.org/dc/elements/1.1/">
6  <body>
7  <a href="" typeof="foaf:Person"
8  about="http://www.theleonardi.com/foaf.rdf" property="foaf:name" >Erwin Leonardi</a>
9
10 <a href="" typeof="swrc:Author"
11 about="http://www.theleonardi.com/foaf.rdf" property="dc:creator" >Erwin Leonardi</a>
12 </body>
13 </html>
```

Fig. 2. Examples of semantic links

Note that conceptually, based on the semantics of the RDFa annotations, there are two triples associated to the link in line 7 of Fig. 2 (expressed here in the notation N3[9]). Fig. 3 represents these triples.



```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/>  .
@prefix dc: <http://purl.org/dc/elements/1.1/> .

<http://www.theleonardi.com/foaf.rdf> rdf:type foaf:Person ;
foaf:name "Erwin Leonardi" .
```

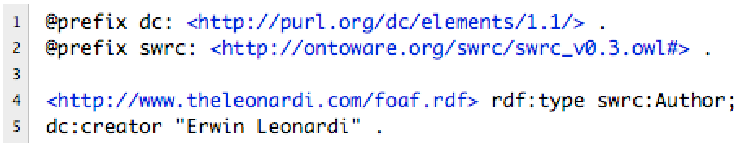
Fig. 3. Example of an HTML link

[7] http://www.w3.org/TR/rdfa-syntax/

[8] http://www.informatik.uni-trier.de/~ley/db/indices/a-tree/l/Leonardi:Erwin.html

[9] http://www.w3.org/2000/10/swap/Primer

Those triples add the following semantics to the link: it is about a person, named "Erwin Leonardi" that is described in the resource *http://www.theleonardi.com/foaf.rdf*. One possible result for this link is to show Erwin Leonardi's Facebook page, since this homepage is a relevant resource semantically related to those triples. Such information could be obtained in Linked Data that describes people or in the resource *http://www.theleonardi.com/foaf.rdf* that describes the person mentioned. On the other hand, for the link in line 10 of Fig. 2, it would be more relevant to show *Erwin Leonardi*'s DBLP publications page, since *author* and *creator* are concepts (see the triples in Fig. 4) that are semantically more related to DBLP[10] Linked Data than the previous mentioned sources.



```
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix swrc: <http://ontoware.org/swrc/swrc_v0.3.owl#> .

<http://www.theleonardi.com/foaf.rdf> rdf:type swrc:Author;
dc:creator "Erwin Leonardi" .
```

Fig. 4. Example of an HTML link

The main benefit of semantic links here is that Linkator can use this extra information for finding a significant value for the *href* attribute, i.e. the link URL, automatically. Therefore, based on vocabularies used for annotating the semantic links, Linkator can select a source on Semantic Web to look for a human-readable representation related to the concept behind the link. This process is further explained in detail in the rest of this paper.

4 Adding Semantic Links

Adding semantic links to a web page means to detect the relevant entities on this page and add anchors that will lead the user to a related document on the web. In this section we will describe these two processes. Fig. 5 illustrates the full process described in this paper.

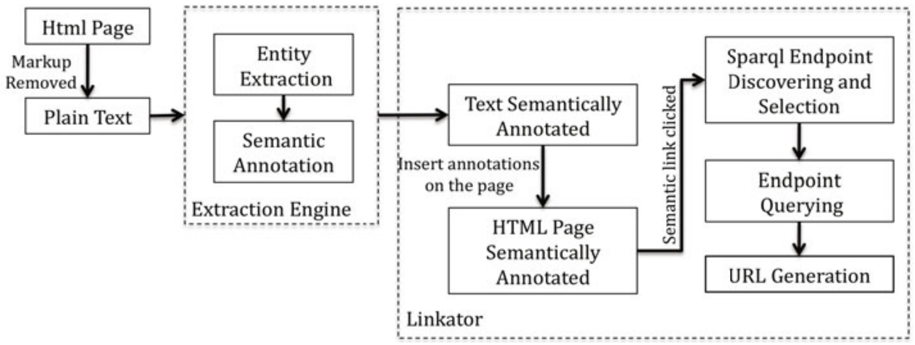
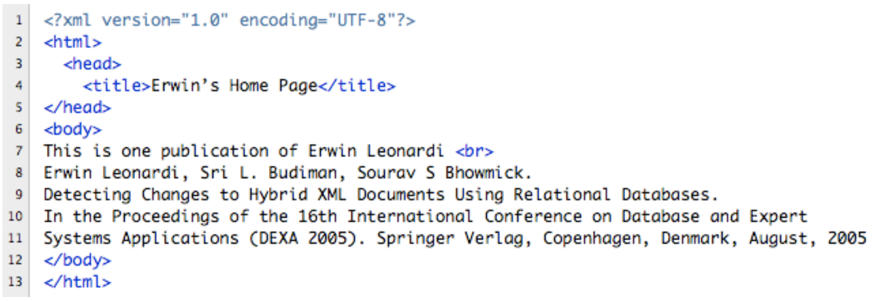


Fig. 5. Semantic link processing flow

[10] `http://dblp.l3s.de/d2r/`

The first stage in the whole process is to detect the relevant entities to be linked. For this, *Linkator* can use any available information extraction engine, due to its flexible and plug & play architecture. The main advantage of this approach is that *Linkator* delegates the intelligence of entity extraction to the extractor tools. However, as most of the extractors do not produce annotations in *RDF* or *RDFa*, a major additional step here is to convert the annotation made by the extractor into *RDFa* annotations. This step is achieved by mapping a specific domain ontology to the schema used by the extractor, and later using that mapping to convert the extractor annotations into *RDFa*. As the result of that entire process, the entities extracted are annotated in a domain specific ontology representing the domain that the extractor was trained for.

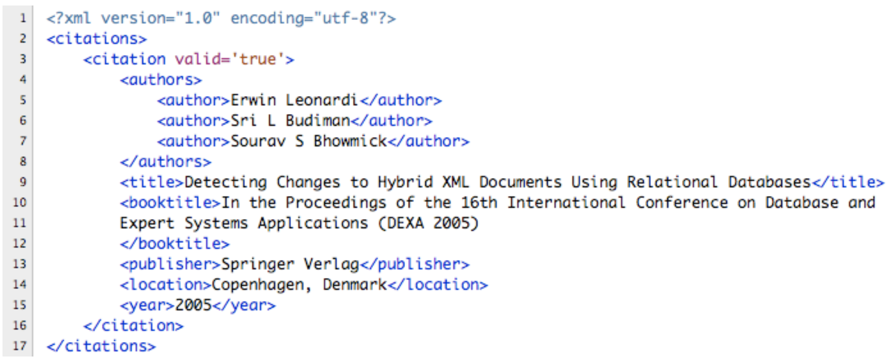
Let us give an example. Suppose that we decided to plug the *Freecite*[11] extractor into *Linkator*. *Freecite* is an entity extraction engine for bibliographic citations. It is able to detect citations in text documents and retrieve a structured XML file containing *authors, title, year of publication, location, number of pages,* and *volume* of the citation.



```
<?xml version="1.0" encoding="UTF-8"?>
<html>
  <head>
    <title>Erwin's Home Page</title>
</head>
<body>
This is one publication of Erwin Leonardi <br>
Erwin Leonardi, Sri L. Budiman, Sourav S Bhowmick.
Detecting Changes to Hybrid XML Documents Using Relational Databases.
In the Proceedings of the 16th International Conference on Database and Expert
Systems Applications (DEXA 2005). Springer Verlag, Copenhagen, Denmark, August, 2005
</body>
</html>
```

Fig. 6. Sample HTML page with a bibliographic reference

Fig. 6 shows an example of a web page with a bibliographic reference. The result of the parsing and extraction of this page in *Freecite* is shown in Fig. 7.



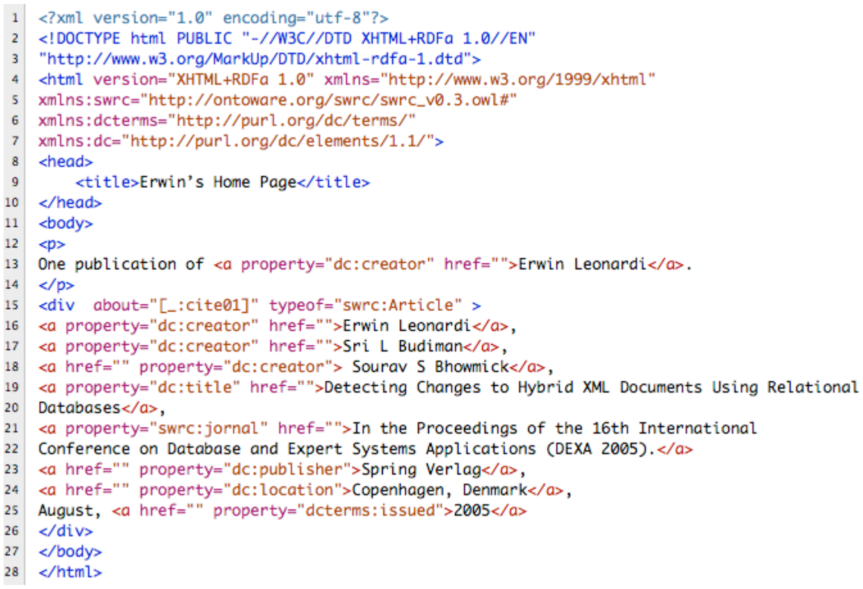
```
<?xml version="1.0" encoding="utf-8"?>
<citations>
    <citation valid='true'>
        <authors>
            <author>Erwin Leonardi</author>
            <author>Sri L Budiman</author>
            <author>Sourav S Bhowmick</author>
        </authors>
        <title>Detecting Changes to Hybrid XML Documents Using Relational Databases</title>
        <booktitle>In the Proceedings of the 16th International Conference on Database and
        Expert Systems Applications (DEXA 2005)
        </booktitle>
        <publisher>Springer Verlag</publisher>
        <location>Copenhagen, Denmark</location>
        <year>2005</year>
    </citation>
</citations>
```

Fig. 7. Output extracted with Freecite in XML

[11] FreeCite - Open Source Citation Parser - freecite.library.brown.edu

Notice that, in particular, Freecite does not directly mark the source page with the corresponding extracted semantic annotation. As was mentioned before, in order to get the original document semantically annotated as output from the extractor, an extended version of *Freecite* was developed in Linkator for directly annotating the original document with *RDFa*. For this purpose, the DBLP ontology[12] (the set of vocabularies used in this dataset) was used as the underlying ontology for the annotation, although any other similar ontology or vocabulary can be used in this process. This Freecite extension is able to transform the Freecite XML output to RDF format and embed the annotations in the original web page. Basically, it transforms the XML file to RDF using an XSL transformation[13]. The resulting file is then automatically injected into the original web page with RDFa annotations. Linkator places the annotation into the original page by matching the literal objects of the RDF triples against the text in the page. The result of this transformation is shown in Fig. 8.

Like this, the first step of the process has been completed as the web page has been extended with semantic links. Yet the computation of the target URL has to be done, and in the next section we choose to do this by exploiting the triples that are associated to the semantic link, which represent the semantics of the link in the page.



```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML+RDFa 1.0//EN"
"http://www.w3.org/MarkUp/DTD/xhtml-rdfa-1.dtd">
<html version="XHTML+RDFa 1.0" xmlns="http://www.w3.org/1999/xhtml"
xmlns:swrc="http://ontoware.org/swrc/swrc_v0.3.owl#"
xmlns:dcterms="http://purl.org/dc/terms/"
xmlns:dc="http://purl.org/dc/elements/1.1/">
<head>
    <title>Erwin's Home Page</title>
</head>
<body>
<p>
One publication of <a property="dc:creator" href="">Erwin Leonardi</a>.
</p>
<div  about="[_:cite01]" typeof="swrc:Article" >
<a property="dc:creator" href="">Erwin Leonardi</a>,
<a property="dc:creator" href="">Sri L Budiman</a>,
<a href="" property="dc:creator"> Sourav S Bhowmick</a>,
<a property="dc:title" href="">Detecting Changes to Hybrid XML Documents Using Relational
Databases</a>,
<a property="swrc:jornal" href="">In the Proceedings of the 16th International
Conference on Database and Expert Systems Applications (DEXA 2005).</a>
<a href="" property="dc:publisher">Spring Verlag</a>,
<a href="" property="dc:location">Copenhagen, Denmark</a>,
August, <a href="" property="dcterms:issued">2005</a>
</div>
</body>
</html>
```

Fig. 8. Original page with semantic links expressed as RDFa annotations

5 Dereferencing Semantic Links

The next step in the process is to define a destination for the added link. Regardless of the destination of the anchor being defined during the annotation process, as

[12] `http://dblp.l3s.de/d2r/`

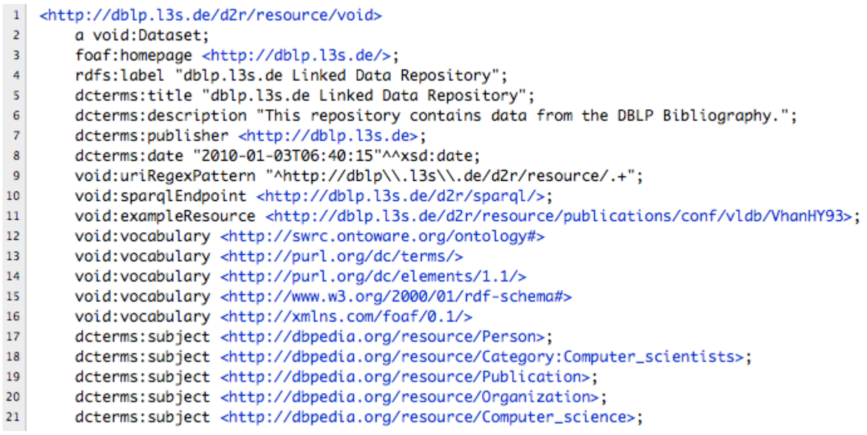
[13] `http://www.w3.org/TR/xslt`

illustrated in [7, 8, 9, 10], in Linkator, it is computed at the moment the user clicks on the *semantic link*. Therefore, it can select the most significant source to find the destination of the anchor, based on the current context where the anchor occurs. To improve performance Linkator uses the Linked Data cloud for discovering destinations for the semantic link as opposed to querying search engines or a fixed knowledge base. For that reason, the destination is computed at the user click, since it would be very slow to query the linked data for find all destination beforehand.

Explained in brief, in order to dereference the semantic link, Linkator queries the Linked Data cloud or an RDF source that exposes a *SPARQL endpoint*, looking for a human-readable representation of the triples related to the semantic link. Thus, the dereferencing process reduces to defining the endpoints to be queried and defining the query itself.

5.1 Endpoint Resolution

SPARQL endpoint resolution is a problem that has recently attracted attention of researchers in the field of federated queries [15, 16, 17, 18, 19, 20]. In spite of the fact that many approaches have been proposed as (partial) solution for this problem there is not yet a consensus or standard. Due to this fact, the Linkator architecture implements its own solution.



```
<http://dblp.l3s.de/d2r/resource/void>
    a void:Dataset;
    foaf:homepage <http://dblp.l3s.de/>;
    rdfs:label "dblp.l3s.de Linked Data Repository";
    dcterms:title "dblp.l3s.de Linked Data Repository";
    dcterms:description "This repository contains data from the DBLP Bibliography.";
    dcterms:publisher <http://dblp.l3s.de>;
    dcterms:date "2010-01-03T06:40:15"^^xsd:date;
    void:uriRegexPattern "^http://dblp\\.l3s\\.de/d2r/resource/.+";
    void:sparqlEndpoint <http://dblp.l3s.de/d2r/sparql/>;
    void:exampleResource <http://dblp.l3s.de/d2r/resource/publications/conf/vldb/VhanHY93>;
    void:vocabulary <http://swrc.ontoware.org/ontology#>
    void:vocabulary <http://purl.org/dc/terms/>
    void:vocabulary <http://purl.org/dc/elements/1.1/>
    void:vocabulary <http://www.w3.org/2000/01/rdf-schema#>
    void:vocabulary <http://xmlns.com/foaf/0.1/>
    dcterms:subject <http://dbpedia.org/resource/Person>;
    dcterms:subject <http://dbpedia.org/resource/Category:Computer_scientists>;
    dcterms:subject <http://dbpedia.org/resource/Publication>;
    dcterms:subject <http://dbpedia.org/resource/Organization>;
    dcterms:subject <http://dbpedia.org/resource/Computer_science>;
```

Fig. 9. Excerpt from the DBLP voID descriptor

In *Linkator*, the *SPARQL endpoint* to be queried is determined based on three things:

1. the triple associated to the semantic link itself,
2. the *RDF* graph of the annotations that exist on the web page,
3. the *voID* (Vocabulary of Interlinked Datasets) [5] descriptors of the available endpoints.

Linkator selects available endpoints based on the vocabularies that they use. The vocabulary that an endpoint uses defines the semantics of the data that it contains to a great extent. Therefore, by matching the vocabulary used on the semantic link with the endpoint vocabularies, *Linkator* can resolve which endpoint may contain significant information about the resources associated with the semantic link. The vocabularies used by the endpoints can be obtained from their *voID* descriptors. *VoID* is an *RDF* vocabulary used to describe linked datasets in the Semantic Web. Fig. 9 shows a fragment of a *voID* descriptor for the *DBLP*[14] endpoint.

When *Linkator* receives a dereferencing request, it executes the algorithm illustrated in Fig. 10 to choose the most relevant endpoint:

```
1   FUNCTION SelectEndpoint
2         E := Array
3         R : = select all rdf:type objects associated to the semantic link
4         T := ExtractVocabulary(R)
5
6         FOR EACH vocabulary in T DO
7         {
8           E.add (select endpoints that contain this vocabulary)
9         }
10        IF E = Empty
11        {
12          R := select all predicates associated to the semantic link
13          T := ExtractVocabulary(R)
14
15          FOR EACH vocabulary in T DO
16          {
17                  E.add (select endpoints that contain this vocabulary)
18          }
19        }
20        RETURN E
21
22  FUNCTION ExtractVocabulary (R)
23        V := Array
24        FOR EACH resource in R DO
25        {
26          V.add (extract the vocabulary from the resource)
27        }
28        RETURN V
```

Fig. 10. Algorithm of the *SelectEndpoint* function that resolves which endpoint will be queried based on the matching between vocabularies.

Let us illustrate how this mechanism works. Suppose that a user clicks on the semantic link *Erwin Leonardi* illustrated in line 16 of Fig. 8. In addition to the triple of this link shown in Fig. 11, all triples annotated and associated to the semantic link can be used during this process.

```
_:cite01 <http://purl.org/dc/elements/1.1/creator> 'Erwin Leonardi'
```

Fig. 11. Triple associated to the semantic link *Erwin Leonardi* in line 16 of Fig. 8

The *SelectEndpoint* function (see Fig. 10, line 3) looks in those triples for an *rdf:type* object. In this example, it matches the resource *swrc:Article* (that represents

[14] The DBLP SPARQL server can be accessed at: `http://dblp.l3s.de/d2r/`

the expanded URL *http://ontoware.org/swrc/swrc_v0.3.owl#Article*). In the next step (Fig. 10, line 4), it extracts the vocabulary associated to this resource. This is done by the function *ExtractVocabulary* (Fig. 10, line 22) that retrieves the vocabulary *http://ontoware.org/swrc/swrc_v0.3.owl,* in this example. The last step in this process is to find the endpoints that use this vocabulary. In line 8, the procedure queries the *voID* descriptor of the available *SPARQL* endpoints, looking for such a vocabulary. The endpoints that contain it will be used during the querying process, as described in the next subsection.

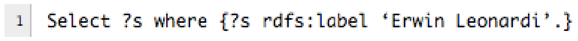
Note that the *SelectEndpoint* function can retrieve more than one endpoint, and in that case, all of them will be used during the querying process. Also, note that in this example, the algorithm found an *rdf:type* object in the annotations. However, in the case where *rdf:type* is not available the engine uses the *predicate* associated to the semantic link, which, in this last example, would be the resource *http://purl.org/dc/elements/1.1/creator*.

In spite of the existence of endpoints covering a broad range of topics (e.g., DBpedia) and using hundreds of vocabularies, most of them are domain-specific and use a small set of vocabularies, which justifies this as a reasonable approach for detecting the endpoint semantically associated to the link being dereferenced.

Nevertheless, several other heuristics can be used to semantically disambiguate the endpoint, for instance, it could be based on the conceptual description of the content of the endpoint. In *voID*, the *dcterms:subject* property should be used to describe the topics covered by the datasets, and in a future version we intent to exploit such information to implement an approach based on concept mapping between the content in endpoint and the semantic links. We also intend to use the voID Store[15] service that aims to aggregate voID descriptors of public endpoints available in the Semantic Web.

5.2 Query Formulation

As mentioned earlier, *Linkator* queries an *RDF* source in order to find a URL for the semantic link. The query itself is created based on the object of the triples associated to the semantic link. The resulting query varies depending on whether the object is a URL (i.e., an *ObjectProperty*) or an *RDF literal* (i.e., a *DatatypeProperty*). In this approach, triples where the object is an *RDF blank node* are discarded. In the case where the object is a literal, the query is *Select ?s where {?s rdfs:label literal}*. For example, for the triple in Fig. 11, one of the *SPARQL* queries generated is shown in Fig. 12.

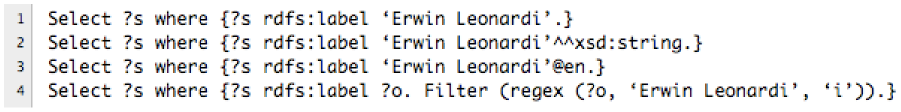


```
Select ?s where {?s rdfs:label 'Erwin Leonardi'.}
```

Fig. 12. Sample query to find URLs for links

[15] http://void.rkbexplorer.com/

In fact, the query exemplified in Fig. 12 is just one of the queries computed. *Linkator* executes a set of queries in order to overcome the limitations posed by the endpoints. For the previous example, the entire set of queries generated is shown in Fig. 13:



```
1 Select ?s where {?s rdfs:label 'Erwin Leonardi'.}
2 Select ?s where {?s rdfs:label 'Erwin Leonardi'^^xsd:string.}
3 Select ?s where {?s rdfs:label 'Erwin Leonardi'@en.}
4 Select ?s where {?s rdfs:label ?o. Filter (regex (?o, 'Erwin Leonardi', 'i')).}
```

Fig. 13. Set of queries generated

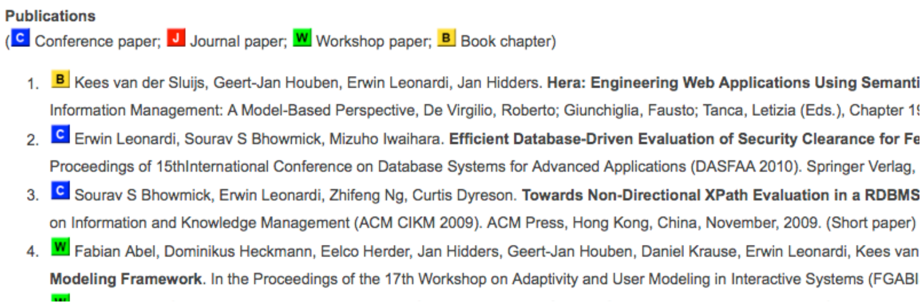
Note however, that the query in line 4 is only executed if the endpoint supports keyword search, otherwise, in practice, such a query has a large probability of timing out or even returning an error. The support for keyword search can be obtained from the *voID* descriptor of the endpoint.

The final step in the dereferencing process is to find a URL to be inserted in the generated XHTML. Since a resource is retrieved in the query, *Linkator* tries to find a URL that contains a human-readable representation associated to the resource. Otherwise, it dereferences the resource URL directly. This same process also applies in the case where the object of the triple is a URI and not a literal. In order to find a human-readable representation for that resource, *Linkator* searches for predicates in the target resource that contain the string 'seealso', 'homepage', 'web' or 'site'. This means that it is able to match predicates such as: *foaf:homepage, akt:has-web-address, rdfs:seeAlso*, that in general, contain a human-readable representation for RDF resources, meaning, a URL for a website. The whole dereferencing process ends with Linkator redirecting the user request to the URL found.

6 Proof of Concept

Linkator has been implemented as an extension to the Firefox browser, together with a backend service used by the extension. In this section, it is illustrated how Linkator can be used for supporting users in their searches where they try to locate PDF files and author's pages for references that they encounter in web pages with citations. In this scenario, the main problem is that some researchers or research groups mention their works on their homepages but they often do not add a link to the referred documents. Therefore, other researchers spend a considerable time looking for copies of the documents on the web: Linkator is able to automatically do this job without any intervention of the user or author. The Linkator prototype used to exemplify this mechanism can be found at: http://www.wis.ewi.tudelft.nl/index.php/linkator

To exemplify this particular scenario, consider Erwin Leonardi's personal homepage: http://www.theleonardi.com/. That page contains a list of Erwin Leonardi's publications, as exhibited in Fig. 14.

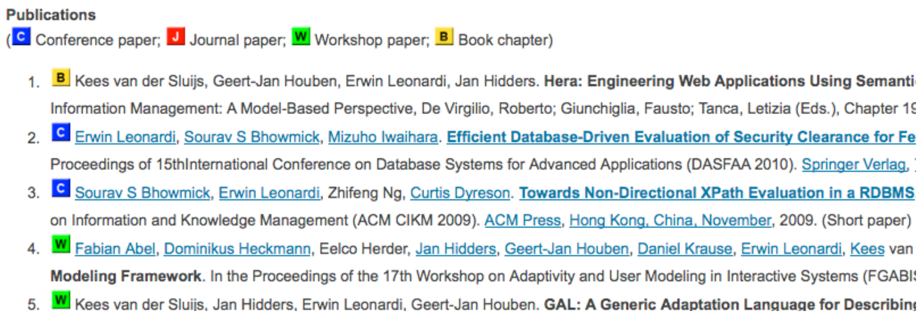


Publications
(C Conference paper; J Journal paper; W Workshop paper; B Book chapter)

1. B Kees van der Sluijs, Geert-Jan Houben, Erwin Leonardi, Jan Hidders. **Hera: Engineering Web Applications Using Semanti**
Information Management: A Model-Based Perspective, De Virgilio, Roberto; Giunchiglia, Fausto; Tanca, Letizia (Eds.), Chapter 1
2. C Erwin Leonardi, Sourav S Bhowmick, Mizuho Iwaihara. **Efficient Database-Driven Evaluation of Security Clearance for Fe**
Proceedings of 15thInternational Conference on Database Systems for Advanced Applications (DASFAA 2010). Springer Verlag,
3. C Sourav S Bhowmick, Erwin Leonardi, Zhifeng Ng, Curtis Dyreson. **Towards Non-Directional XPath Evaluation in a RDBMS**
on Information and Knowledge Management (ACM CIKM 2009). ACM Press, Hong Kong, China, November, 2009. (Short paper)
4. W Fabian Abel, Dominikus Heckmann, Eelco Herder, Jan Hidders, Geert-Jan Houben, Daniel Krause, Erwin Leonardi, Kees van
Modeling Framework. In the Proceedings of the 17th Workshop on Adaptivity and User Modeling in Interactive Systems (FGABI

Fig. 14. Excerpt from Erwin Leonardi's publications listed in http://www.theleonardi.com

The same page, after it has been (automatically) processed by Linkator, is shown in Fig. 15. The semantic links added by Linkator are shown in blue.



Publications
(C Conference paper; J Journal paper; W Workshop paper; B Book chapter)

1. B Kees van der Sluijs, Geert-Jan Houben, Erwin Leonardi, Jan Hidders. **Hera: Engineering Web Applications Using Semanti**
Information Management: A Model-Based Perspective, De Virgilio, Roberto; Giunchiglia, Fausto; Tanca, Letizia (Eds.), Chapter 1
2. C Erwin Leonardi, Sourav S Bhowmick, Mizuho Iwaihara. **Efficient Database-Driven Evaluation of Security Clearance for Fe**
Proceedings of 15thInternational Conference on Database Systems for Advanced Applications (DASFAA 2010). Springer Verlag,
3. C Sourav S Bhowmick, Erwin Leonardi, Zhifeng Ng, Curtis Dyreson. **Towards Non-Directional XPath Evaluation in a RDBMS**
on Information and Knowledge Management (ACM CIKM 2009). ACM Press, Hong Kong, China, November, 2009. (Short paper)
4. W Fabian Abel, Dominikus Heckmann, Eelco Herder, Jan Hidders, Geert-Jan Houben, Daniel Krause, Erwin Leonardi, Kees van
Modeling Framework. In the Proceedings of the 17th Workshop on Adaptivity and User Modeling in Interactive Systems (FGABI
5. W Kees van der Sluijs, Jan Hidders, Erwin Leonardi, Geert-Jan Houben. **GAL: A Generic Adaptation Language for Describin**

Fig. 15. Excerpt from Erwin Leonardi's publications enriched with semantic links

After Linkator processes the page, clicking on the name of an author (annotated with *dc:creator*) makes Linkator search for a human-readable representation of the author in the DBLP SPARQL endpoint, as described in the previous section. As a result, it retrieves a homepage of this author, as registered in the DBLP records. For instance, by clicking on "Jan Hidders" the homepage http://www.wis.ewi.tudelft.nl/index.php/personal-home-page-hidders is retrieved, while by clicking on "Geert-Jan Houben" the homepage http://wwwis.win.tue.nl/~houben/ is retrieved, since both URLs are stored in the DBLP endpoint. The same applies to the title of the citations that were annotated by Linkator with semantic links. By clicking on a title (annotated with *dc:title*), Linkator searches in DBLP for a human-readable representation of this title. For instance, in Erwin Leonardi's page, by clicking in the title "XANADUE: A System for Detecting Changes to XML Data in Tree-Unaware Relational Databases", it retrieves the ACM page of this article (http://portal.acm.org/citation.cfm?doid=1247480.1247633). In this page the user can find the PDF file for the article.

This proof of concept scenario shows how Linkator operates. Although the example solves a trivial task, the full approach exemplifies how the problem of auto-linking page can be decomposed to support a domain-independent and an adaptive approach.

7 Conclusion and Future Work

Linkator is an architecture that brings the benefits of semantic annotation closer to end-users. It automatically adds links to web pages, which increases the connectivity of the web page with related external resources, consequently improving user navigation and access to web resources. This paper shows how this auto-linking problem can be solved by incorporating elements of the Semantic Web into a solution based on semantic links. Also, it focuses on the process of finding a URL for the semantic link, using the Semantic Web as underlying knowledge base. Linkator is able to select a source to be queried based on the semantics of the annotations on the page. Therefore, it can determine the URL of the link, by exploiting a knowledge base that is semantically related to the concept that is being linked. The Linkator prototype exemplifies how knowledge workers can benefit from such a mechanism to find documents related to bibliographic citations mentioned in web pages.

As future work, we intend to extend and measure Linkator's discovery and query model for improving the dereferencing mechanism; investigate the use of more general adaptation and recommendation approaches that can be enriched with Linkator's added semantics; improve the Linkator resolution mechanism when the endpoint retrieves more than one human-representation for an semantic link; and investigate how the link generation can benefit from integrating with classical search engines as well.

References

1. Alexander, K., Cyganiak, R., Hausenblas, M., Zhao, J.: Describing Linked Datasets - On the Design and Usage of voiD, the 'Vocabulary of Interlinked Datasets'. In: Linked Data on the Web Workshop (LDOW '09), in conjunction with 18th International World Wide Web Conference, WWW '09 (2009)
2. Hughes, G., Carr, L.: Microsoft Smart Tags: Support, ignore or condemn them? In: Proceedings of the ACM Hypertext 2002 Conference, Maryland, USA, pp. 80–81 (2002)
3. Medelyan, O., Witten, I.H., Milne, D.: Topic Indexing with Wikipedia. In: Proceedings of the AAAI 2008 Workshop on Wikipedia and Artificial Intelligence (WIKIAI 2008), Chicago, IL (2008)
4. Mihalcea, R., Csomai, A.: Wikify!: linking documents to encyclopedic knowledge. In: Proceedings of the 16th ACM Conference on Information and Knowledge management (CIKM '07), Lisbon, Portugal, pp. 233–242 (2007)
5. Milne, D., Witten, I.H.: Learning to link with wikipedia. In: Proceeding of the 17th ACM conference on Information and knowledge management, Napa Valley, California, USA, October 26-30 (2008)
6. Gardner, J., Xiong, L.: Automatic Link Detection: A Sequence Labeling Approach. In: International Conference on Information and Knowledge Management, CIKM '09 (2009)
7. Reeve, L., Han, H.: Survey of semantic annotation platforms. In: Proceedings of the 2005 ACM Symposium on Applied computing, Santa Fe, New Mexico, March 13-17 (2005)
8. Hartig, O., Bizer, C., Freytag, J.: Executing SPARQL Queries over the Web of Linked Data. In: Bernstein, A., Karger, D.R., Heath, T., Feigenbaum, L., Maynard, D., Motta, E., Thirunarayan, K. (eds.) ISWC 2009. LNCS, vol. 5823, pp. 293–309. Springer, Heidelberg (2009)

9. Schenk, S., Staab, S.: Networked graphs: a declarative mechanism for SPARQL rules, SPARQL views and RDF data integration on the web. In: Proceeding of the 17th international conference on World Wide Web (WWW '08), New York, NY, USA, pp. 585–594 (2008)
10. Harth, A., Hose, K., Karnstedt, M., et al.: On Lightweight Data Summaries for Optimised Query Processing over Linked Data (2009)
11. Zemanek, J., Schenk, S., Svatek, V.: Optimizing SPARQL Queries over Disparate RDF Data Sources through Distributed Semi-Joins. In: ISWC 2008 Poster and Demo Session Proceedings. CEUR-WS (2008)
12. Bouquet, P., Ghidini, C., Serafini, L.: Querying the Web of Data: A Formal Approach. In: The Semantic Web, pp. 291–305 (2008)
13. Langegger, A., Wöß, W., Blöchl, M.: A Semantic Web Middleware for Virtual Data Integration on the Web. In: Bechhofer, S., Hauswirth, M., Hoffmann, J., Koubarakis, M. (eds.) ESWC 2008. LNCS, vol. 5021, pp. 493–507. Springer, Heidelberg (2008)
14. Gardner, J.J., Krowne, A., Xiong, L.: NNexus: An Automatic Linker for Collaborative Web-Based Corpora. IEEE Trans. Knowl. Data Eng. 21(6), 829–839 (2009)