Building a Semantic Web Search Engine: Challenges and Solutions

Andreas Harth, Aidan Hogan, Jürgen Umbrich, Stefan Decker

Digital Enterprise Research Institute (DERI)

National University of Ireland, Galway

Abstract

Current web search engines return links to documents for user-specified keywords queries. Users have to then manually trawl through lists of links and glean the required information from documents. In contrast, semantic search engines allow more expressive queries over information integrated from multiple sources, and return specific information about entities, for example people, locations, news items. An entity-centric data model furthermore permits powerful query and browsing techniques. In this paper, we report on our experiences in collecting and integrating Web data from millions of sources, and describe both application-developer query services and end-user navigation services offered by SWSE, the Semantic Web Search Engine.

Introduction

Traditional Web search engines such as Google, Yahoo, Live Search, and Ask have been designed and optimised for locating documents online, and they perform well at returning documents relevant to specified keywords. The interaction model is simple yet effective: users specify keywords in an input field

and the search engine returns links to the top ten Web documents matching the keywords.

The document-centric model, however, is ill-suited for more complex information foraging tasks which require structured information integrated from a multitude of sources. The entity-centric model presented here allows for describing entities such as people, locations, or news items, rather than just documents, and allows application developers to write software programs that leverage the underlying dataset with unprecedented ease and scale.

The utility of any search engine depends on two parts: the quality of the system, and content, which in our case is provided by a large number of contributors (personal and corporate web sites, for example). Importantly, content suppliers have to agree on a social contract (as anywhere on the Internet) on how to provide and publish data. One goal of this paper is to report on how RFCs, W3C recommendations, and the following of best practices facilitate data reuse.

In addition to data provisioning, there are a number of challenges in implementing a Semantic Web search engine:

• The architecture of a semantic search engine must scale to the Web.

• Dealing with data rather than documents requires a different indexing approach compared to traditional information retrieval systems.

• Data from the Web is of varying quality, which poses challenges for data cleansing and entity consolidation.

• The schema of the data is not known a priori, which makes building user interfaces difficult.

In the following we give an overview of the architecture of SWSE, a search engine that scales to billions of RDF statements; and discuss the necessary adaptations to traditional search engine components, mainly targeting the data acquisition and processing phases. We describe both end-user navigation services and application-developer query services offered by SWSE.

System Overview and Architecture

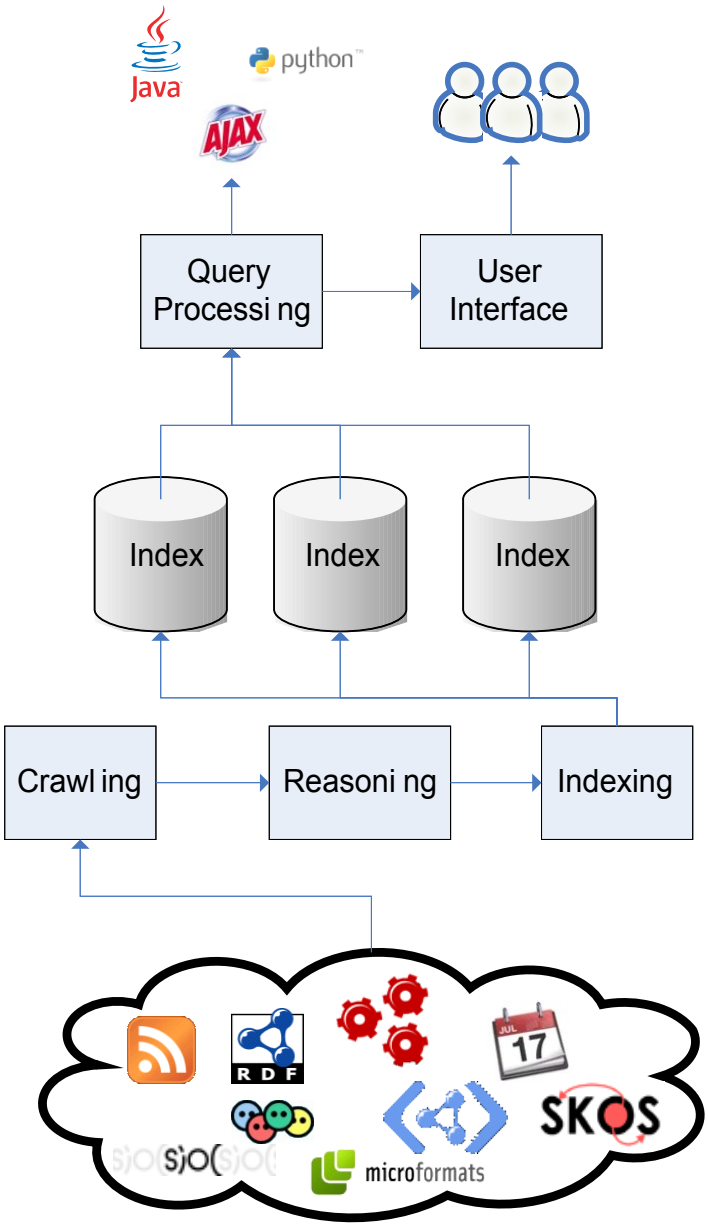
Here we give an overview of the SWSE architecture with particular focus on the data processing pipeline which involves the following steps:

- The crawler gathers data from the Web by traversing the link graph and transforms metadata from HTML documents (e.g. RDFa, GRDDL, or Microformats) and metadata embedded in various file formats (e.g. PDF, PNG, MS Office) into RDF. The crawler also extracts RDF from RSS 2. 0 and Atom feeds.

- Reasoning is implemented to improve the quality of data, create new relationships between entities in the data, and perhaps most importantly, to merge data from multiple sources and schemas into a consolidated dataset. Reasoning is used by exploiting OWL [*SWM04*] and RDFS descriptions of a given domain to infer new knowledge about instances in that domain.

- SWSE supports SPARQL [*PS08*], a W3C Recommendation for an RDF query language. The index structure comprises a complete index on quadruples [*HD05*] with keyword search functionality based on a standard inverted index. The index and query processing components can be distributed across a number of machines [*HUHD07*].

The process of collecting and preparing data to allow for the provisioning of query and navigation services is illustrated in Figure 1.



swse_arch.png

Figure 1. SWSE architecture and data flow overview.

API and User Interface

The current index of SWSE contains data collected from millions of sources. As a running example, we use data from the XTech website and the website of Dan Brickley, the creator of FOAF, to illustrate how SWSE collects and integrates data and provides access to the dataset for both human users and software agents.

Software developers can write programs or widgets that query the SWSE API and reuse or simply render the resulting data. For example, the query in Figure 2 returns the name, nickname and homepage (if available) of people that Dan Brickley knows. The result of the query is shown in Table 1. Please note that results can be returned in JSON format to facilitate processing in JavaScript applications.

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>

SELECT DISTINCT ?name ?nick ?homepage

WHERE {

 <http://danbri.org/foaf.rdf#danbri> foaf:knows ?person .

 ?person foaf:name ?name .

 OPTIONAL { ?person foaf:nick ?nick . }

 OPTIONAL { ?person foaf:homepage ?homepage . }

}
```

Figure 2: SPARQL query for name, nickname, and homepage of acquaintances of Dan Brickley.

name	**nick**	**homepage**
Tim Berners-Lee		http://www.w3.org/People/Berners-Lee/
Jim Ley	JibberJim	
Edd Dumbill	Edd	http://heddley.com/edd/
Martin Poulter		http://www.weird.co.uk/martin/
Libby Miller		
Amy van der Hiel		
Joe Brickley		
Eric Miller		http://purl.org/net/eric/
Jan Grant		

Aaron Swartz		
Dave Beckett		
Art Barstow		
Dan Connolly	DanCon	
Damian Steer	damey	
Ludovic Hirlimann	Softkid	http://perso.hirlimann.net/~ludo/
Tatiana de la O	acracia	http://delcorp.org/abbadingo
Dean Jackson		http://www.grorg.org/dean/
Mr Benn		http://www.mrbenn.co.uk/

Table 1: Name, nickname, and homepage of acquaintances of Dan Brickley.

A user interface for RDF data has to balance two opposing requirements: being easy to use and allowing for complex query operations. Additionally, the user interface should be as schema-independent as possible; on the Web we might come across thousands of predicates and classes and we are not able to supply displaying templates for all of them. A domain-independent UI has thus to operate on a best-effort basis and display data even if the schema is not known a priori. For a state-of-the-art survey of semantic search interfaces see [*HOH07*].

The user interface currently supports three types of queries: 1) keyword search, 2) restriction by type, 3) entity browsing. Keyword search can be done without knowledge of the schema. Once users have narrowed in on a set of keyword-matching entities, they can restrict the result set further by specifying the type of result they require (e.g. Person). In the next step, users can view details about a selected entity and can navigate along links to associated entities. For an example of an entity view see Figure 3 which displays the information about Edd Dumbill available in the system.

edd.png

Figure 3: Entity view of Edd Dumbill, displaying data integrated from four different sources and augmented with reasoning results.

While conducting user experiments we found that the concept of directionality of connections between entities is difficult to communicate to users; hence, we use reasoning (in particular `owl:inverseOf` and `owl:SymmetricProperty`) to materialise links in both directions.

Data Acquisition and Pre-processing

In this section we explain how SWSE processes and transforms data from different formats to RDF. We use a modified version of the crawling framework MultiCrawler [*HUD06*].

In the first step, the web crawler downloads the content from a seed set of URIs and extracts links from HTML and RDF documents for the next crawling round. A link in an RDF document is described using the property `rdfs:seeAlso`. Figure 4 shows an example of an RDF link, referring from Dan Brickley's FOAF URI to his social bookmark RSS feed.

```
<foaf:Person rdf:about=http://danbri.org/foaf.rdf#danbri>
 <rdfs:seeAlso rdf:resource="http://del.icio.us/rss/danbri">
 ….
</foaf:Person>
```

Figure 4: rdfs:seeAlso link from the foaf:Person entity to his social bookmark feed

Apart from parsing links from common HTML links (`a[@href]` and `img[@src]`) we also extract URIs from eleven different HTML tags (e.g. `frame[@src]`, `link[@href]` or `object[@src]`).

Autodetection links in the `html\head` tag give authors the possibility to support data aggregators in discovering URIs to their data (see Figure 5).

```
<head>
 <link rel="meta" type="application/rdf+xml" title="FOAF"
 href="http://danbri.org/foaf.rdf" />
 <link rel="alternate" type="application/rss+xml"
 title="RSS1.0"
 href="http://feeds.feedburner.com/danbri_blog?format=xml"/>
```

Figure 5: Autodection link in the html\head element to the FOAF file and an RSS feed.

To supply the reasoning process with OWL and RDFS descriptions, the crawler downloads the data pointed to by namespace URIs. The list of namespace URIs are extracted from either `head[@profile]` HTML attributes (e.g. first line in Figure 8) or from the RDF data.

Once the crawler has fetched the data from the Web, we process the content and extract embedded information, such as RDFa and Microformats, and convert it into RDF. Figure 6 shows an example of embedded RDFa in Dan Brickley's HTML homepage and Figure 7 shows the related RDF output after the extraction process. An example of the use of Microformats in HTML can be found at the XTech conference schedule page (Figure 8 shows a code snippet).

```
<span rel="foaf:depiction">
 <img src="danbri-txt.jpg" alt="danbri" style="float:
center" property="foaf:sha1"
     content="58d174f20c039289544b2364c5c21295df2e4a2b"/>
</span>
```

Figure 6: Embedding RDFa in HTML

```
<foaf:Document rdf:about="http://danbri.org/">
 <foaf:depiction>
  <foaf:Image rdf:about="http://danbri.org/danbri-
txt.jpg">
   <foaf:sha1>58d174f20c039289544b2364c5c21295df2e4a2b</fo
af:sha1>
   …
  </foaf:Image>
 </foaf:depiction>
 …
</foaf:document>
```

Figure 7: RDF output using http://www.w3.org/2007/08/pyRdfa/

```
<head profile="http://www.w3.org/2002/12/cal/hcal
http://www.w3.org/2006/03/hcard">

...

 <div class="slot topic23" id="slot419"><div
class="slot_detail vevent">

  <abbr class="dtstart" title="20080509T0900"></abbr>

  <abbr class="dtend" title="20080509T0945"></abbr>

  <span class="summary">

   <a href="/public/schedule/detail/477" class="url uid">

  Building a Semantic Web Search Engine: Challenges and
Solutions

   </a>
```

```
  </span>
  <span class="description">Aidan Hogan (DERI Galway)</span>
 </div>
```

Figure 8: Use of Microformats in the schedule page of the XTech conference.

The SWSE crawler also supports the transformation of iCal documents into RDF. For other file formats we use various Python or C libraries to extract and convert the embedded meta information into RDF. For example WORD, EXCEL or PDF documents and various audio, video or image formats contain information about the author, creation time, a keyword/tag list, content type, and encoding. The crawler is also able to exploit data from databases which has been exported to the Web via D2R[1] and Triplify[2], taking into account sitemap extensions[3]. For a more comprehensive sitemap specification see [*CDSTD08*].

In the last pre-processing step we cleanse the RDF output and fix character encoding issues, check for valid URIs and canonicalise them (e.g. adding trailing "/"), or remove XML tags and JavaScript code from the textual content of the data.

Reasoning

We now examine the results of performing reasoning on the data produced by the above data acquisition and pre-processing.

We currently implement a scalable reasoning system which can operate over large volumes of data. The reasoning engine is equipped to support a significant subset of OWL. For the purposes of this paper, we will demonstrate the precise advantages of reasoning by examining data before and after reasoning; for clarity, we examine only the most useful and most recognisable reasoning entailments. Firstly, we examine a possible equality reasoning scenario [*HHD07*]; secondly, we detail the benefit of generic reasoning in creating new knowledge and

---

[1] http://www4.wiwiss.fu-berlin.de/bizer/d2rmap/D2Rmap.htm

[2] http://triplify.org/

[3] http://sitemap.org/

materialising new relations between entities.

On the Web, it is quite common for different sources to contribute knowledge about the same entity but under different identifiers. To illustrate, prior to reasoning on the running example dataset, there were ten entities with value "Dan Brickley" for `foaf:name`: three of these were URIs (see Table 2) and seven were blank node identifiers . These entities were described in different sources such as Dan Brickleys FOAF file, LiveJournal, MyOpera etc. Pre-reasoning, each entity is treated within the SWSE architecture as a separate result although we know in this case that it is likely the same person is being described.

`<http://danbri.org/foaf.rdf#danbri>`
`<http://danbri.org/foaf#danbri>`
`<http://my.opera.com/danbri/xml/foaf#me>`

Table 2: List of three URIs used to identify Dan Brickley

Reasoning performs matching on properties defined in their ontology as one which uniquely describes an entity; such properties are said to be a member of the `owl:InverseFunctionalProperty` class. Figure 8 lists three equivalent entities, referring to Dan Brickley, found in different sources, which will be consolidated through the inverse functional properties `foaf:mbox_sha1sum` and `foaf:homepage`

```
<foaf:Person rdf:about="http://danbri.org/foaf.rdf#danbri
">
<foaf:mbox_sha1sum>6e80d02de4cb3376605a34976e31188bb16180d
0</foaf:mbox:sha1sum>
<foaf:homepage rdf:resource="http://danbri.org/" />
...
</foaf:Person>


<foaf:Person>
<foaf:mbox_sha1sum>6e80d02de4cb3376605a34976e31188bb16180d
0</foaf:mbox:sha1sum>
...
</foaf:Person>
```

```
<foaf:Person>
<foaf:homepage rdf:resource="http://danbri.org/" />
...
</foaf:Person>
```

Figure 8: Three equivalent entities describing Dan Brickley, including two anonymous entities.

After reasoning, the running example contains three entities with `foaf:name` "Dan Brickley". Two of the entities are in fact the same person, but no match was found on values for inverse-functional properties. The other entity is a different "Dan Brickley", described by his namesake (the protagonist of this running example) in his FOAF file.

Thus far, we have only examined a subset of reasoning useful for consolidating entity descriptions derived from multiple sources. Reasoning can also be used to enrich a knowledge-base with inferred statements which would otherwise not be available for servicing queries. We will now motivate reasoning by illustrating some inferencing results based on the entailment of property definitions.

In OWL ontologies, properties are often described in terms of other properties. For example, properties may extend other properties, be defined as symmetric or have an inverse property defined. Statements which use such properties to describe an entity can then be used to infer more statements.

For example, the SWSE user interface ideally requires an `rdfs:label` value for each entity, the value of which is a human readable name or summary of the entity. However, many ontologies describe their own specific properties with similar semantics as the `rdfs:label`, examples include `foaf:name`, `rss:title`, `dc:title`, etc. Most ontologies define such properties as extending `rdfs:label` using the `owl:subPropertyOf` construct. Thus for example, using reasoning, we can say that an entity which has `foaf:name` "Dan Brickley" also has `rdfs:label` "Dan Brickley". Also, materialising a property hierarchy can be useful to allow users be as specific or as general as they require in searching or navigating between entities.

Inverse properties and symmetric properties specifically increase the linkage of the dataset and offer new navigation paths between entities in the data. For example, the `foaf:made` property is used to link people to documents they authored. In the FOAF specification, another property, `foaf:maker`, is specified

as being the inverse of `foaf:made` and linking documents to their authors. Reasoning is used to materialise all possible relations computable from `owl:inverseOf` relations. Symmetric properties are those which are the inverse of themselves: also known as bi-directional or directionless relations. Reasoning is used to ensure that the relation is asserted in both directions. Materialising relations allows for servicing more queries given the same data, the benefits of which are propagated to human users and agents.

Conclusion

In this paper we have presented the architecture and implementation of SWSE, a semantic web search engine. We have illustrated the data gathering and integration process of SWSE over open web data. The current demo featuring a dataset collected from millions of sources is online at http://swse.deri.org/. Access to the SPARQL endpoint is provided at http://swse.deri.org/yars2/.

References

[CDSTD08] Richard Cyganiak, Renaud Delbru, Holger Stenzhorn, Giovanni Tummarello and Stefan Decker. “Semantic Sitemaps: Efficient and Flexible Access to Datasets on the Semantic Web”. 5th European Semantic Web Conference, Tenerife, Spain, June 1 to 5, 2008.

[HD05] Andreas Harth, Stefan Decker. "Optimized Index Structures for Querying RDF from the Web". 3rd Latin American Web Congress, Buenos Aires, Argentina, October 31 to November 2, 2005, pp. 71-80.

[HHD07] Aidan Hogan, Andreas Harth, Stefan Decker. "Performing Object Consolidation on the Semantic Web Data Graph". Proceedings of I3: Identity, Identifiers, Identification. Workshop at 16th International World Wide Web Conference (WWW2007), Banff, Alberta, Canada, 2007.

[HOH07] Michiel Hildebrand, Jacco van Ossebruggen, Lynda Hardman, “An Analysis of Search-based User Interaction on the Semantic Web”, CWI Technical Report, 2007.

[HUD06] Andreas Harth, Jürgen Umbrich, Stefan Decker. "MultiCrawler: A

Pipelined Architecture for Crawling and Indexing Semantic Web Data". 5th International Semantic Web Conference, Athens, GA, USA. November 5-9, 2006.

[HUHD07] Andreas Harth, Juergen Umbrich, Aidan Hogan, Stefan Decker. "YARS2: A Federated Repository for Querying Graph Structured Data from the Web". 6th International Semantic Web Conference, Busan, Korea, November 11-15, 2007.

[PS08] Eric Prud'hommeaux, Andy Seaborne. “SPARQL Query Language for RDF”, W3C Recommendation, January 15, 2008. http://www.w3.org/TR/rdf-sparql-query/

[SWM04] Michael K. Smith, Chris Welty, Deborah McGuiness, “OWL Web Ontology Language Guide”, W3C Recommendation, February 10, 2004. http://www.w3.org/TR/owl-guide/