Squiggle: An Experience in Model-Driven Development of Real-World Semantic Search Engines

Irene Celino, Emanuele Della Valle, Dario Cerizza, and Andrea Turati

CEFRIEL – Politecnico of Milano, Via Fucini 2, 20133 Milano, Italy
{celino,dellavalle,cerizza,turati}@cefriel.it

Abstract. Search engines are becoming such an easy way to find textual resources that we wish to use them also for multimedia content; however, syntactic techniques, even if promising, are not up to the task: future search engines must consider new approaches. In order to prove that Semantic Web technologies provide real benefits to end users in terms of an easier and more effective access to information, we designed and developed Squiggle, a Semantic Web framework that eases the deployment of semantic search engines. Following a model-driven approach to application development, Squiggle makes ontologies part of the running code. We evaluate the advantages of Squiggle against traditional approaches in real world deployments.

1 Introduction

Searching everything everywhere is becoming our habit when we need to find something. We search Web pages in Web search engines, music using search engines integrated in multimedia players, pictures in images organizer applications, even personal stuff using desktop searches. However, *finding* what we need is often a hard job. Current search engine technology is very good in finding complete Web pages published all over the world, but it lacks the desired precision[1] and recall[2] when searching for multimedia resources. For instance, searching "jaguar" in an image search engine results in a mix of felines and cars, which are difficult to tell apart. Moreover, current technology is unable to cope with results that requires either to extract a part of a resource (e.g., a scene from a movie) or to aggregate numerous resources (e.g., relevant but scattered information regarding a person).

Furthermore, searching is an *expensive activity*. For instance, in a medium-sized enterprise with 100 employees, each one of them would perform around 10 searches per day (some on the Web, some on their mailboxes, etc.), stopping, successfully or not, in 1-2 minutes. This means that 20-30 hours a day are spent in searching.

[1] Precision is the proportion of relevant data of all data retrieved.
[2] Recall is the proportion of retrieved relevant data, out of all available relevant data.

L. Baresi, P. Fraternali, and G.-J. Houben (Eds.): ICWE 2007, LNCS 4607, pp. 485–490, 2007.

What we really need is a search engine able to find any kind of multimedia resource with the required level of granularity; but, how can we achieve this *search engine of the future*? We believe that Tim Berners-Lee was right when, drawing the "Semantic Web Roadmap"[1], he said:

> If an engine of the future combines a reasoning engine with a search engine, it may be able to get the best of both worlds.

Keeping in mind Tim Berners-Lee claim, we conceived, implemented and deployed Squiggle (`http://squiggle.cefriel.it`) an extensible semantic search framework designed to add a conceptual flavour at indexing time and to exploit as much as possible ontological elements to improve searching time.

2 Existing Approaches to Improve Search Engines

A standard "syntactic" search engine's implementation is mainly based on three phases [2]: (a) *crawling time*, the phase in which the resources are collected in order to build a coherent (and more homogeneous) set; (b) *indexing time*, the phase during which the crawled resources are parsed and indexed in some particular data structures, optimized to quickly answer to queries; (c) *searching time*, the run-time phase in which final users submit their queries in order to retrieve meaningful results, possibly ranked and/or clustered.

When the resources to be indexed are multimedia files instead of Web pages, the automatic process of their content becomes very difficult and the lack of links makes crawling a tricky problem and Google PageRank algorithm useless. The two ways out are the use of smart machines and smart data. By *smart machine* we mean a bunch of techniques that includes text processing, audio processing and image/video processing. Several search engines that exploit smart machines are appearing (e.g., Retriever or Musipedia[3]). On the other side, *smart data* is the base for search engines that exploits semantics at search time to increase both recall and precision. I.e., Semantic Web standards offer the possibility of modeling the domain both at lexical and at knowledge level. Explicit representation of semantics gives search engines the ability to disambiguate between homonyms and expand the search to synonyms, pseudonyms and any other relation.

There are several examples of existing approaches that try to *combine Semantic Web technologies with smart machines* in search engines. One of the most interesting is represented by KIM [3], which includes a semantically enhanced information extraction system, which provides automatic semantic annotation with references to classes and instances in the ontology.

3 Our Steps Towards the "Search Engine of the Future"

In our opinion, what Tim Berners-Lee calls the "search engine of the future" should have a structure similar to existing "syntactic" search engines, but should

[3] `http://labs.systemone.at/retrievr` and `http://www.musipedia.org/`

also be enriched with machine-processable semantics. In our vision, domain ontologies can be employed in empowering searching, indexing and also crawling.

At *crawling time*, a previous knowledge about the domain can assist the collecting of resources, because this "know-how" can drive the crawler to focus on relevant information even if links are not explicit. At *indexing time*, the input information can be analyzed by means of smart machines and tagged with respect to its meaning before it is processed by the indexer tool. In this way, the tool is able to index both the syntactic content of an input document and its attached semantics. At *searching time*, domain ontologies can be employed to customize search engine applications and to improve the user experience in terms of value added and effectiveness of the search. The tool can help the user in refining his query both by clarifying the matter of his search and by suggesting possible expansions of his query to related subjects. The result is that the user can find more easily what he was looking for.

We conceived Squiggle, keeping in mind Tim Berners-Lee claim and the above analysis. Squiggle is an extensible framework designed to add a conceptual layer to indexing process and to exploit as much as possible ontological elements to improve searching time, leaving to each domain-dependent instantiation of the framework the choice of using ontologies also at crawling time.

4 Conceptual Architecture of the Squiggle Framework

As a result of our analysis and design, Squiggle is:

- a *semantic search-engine*, i.e. a semantic web application with searching functionalities; and
- a *semantic-search engine*, i.e. a search engine that is able to deal with the "meaning" of the searched information.

While the latter objective concerns the semantics of the *data* and can be achieved through an opportune modeling of knowledge, the former purpose is strictly related to the model-driven development of a semantic web *application*.

Squiggle is indeed a semantic application, since its design heavily grounds on a common model, provided by SKOS [4], which permeates all its structure; Squiggle assumes SKOS as its application model and, therefore, is naturally able to exploit SKOS' semantics. Moreover, being SKOS a horizontal ontology (therefore not bound to any specific subject), Squiggle is completely domain-independent and can thus serve as a *framework* to build domain-specific search applications. In essence, Squiggle is not a search engine itself, but it allows users to customize their own engine on the basis of a particular domain knowledge.

Squiggle is designed to provide both syntactic and semantic indexing and searching primitives, seamlessly combining the speed of syntactic search tools with improved recall and precision, based on the ability to assign alternative designations and wordings in multiple languages to their meaning. Among the constituents of Squiggle, Sesame [5] is used as the semantic engine that queries the knowledge base, whereas the syntactic search engine Lucene [6] is used to

quickly perform text searches. Therefore the described architecture lends itself well both to overcome the limitations of purely syntactic approaches and to improve the performance of semantic engines.

Technically speaking, Squiggle's innovation consists in its Conceptual Indexing and Semantic Search capabilities. The *Conceptual Indexing* consists of a *semantic annotation process*, during which the input information is scanned and analyzed in order to identify and extract the concepts that characterize it (Squiggle expects resources to be annotated with keywords and it searches in the domain ontology for concepts whose SKOS labels match those keywords), and of an *indexing process*, during which these concepts are stored in an index for subsequent search and retrieval. On the other hand, the *Semantic Search* analyzes user' queries and tries to identify the ontological elements that can be related to the request, "suggesting" to the user the potential *meanings* of his query; the user is therefore presented with both the results of the syntactic search and the available meanings extracted from the query, which can help him to refine his request, "disambiguating" among its the possible acceptations. Moreover, when a user query is re-conducted to a specific meaning, Squiggle also seeks other concepts that could be of interest for the user; this is possible because Squiggle Semantic Search can navigate across the graph of interconnected elements of the domain ontology, following "semantic paths" denoted by relations and attributes.

5 Squiggle Real-World Deployments

In order to prove the feasibility of our approach, we briefly present some test beds. We successfully developed some search engines on top of the Squiggle framework, in different application fields.

Squiggle Ski – CEFRIEL, as Official Supplier of the XX Olympic Winter Games for Applied Academic Research, caught the opportunity to demonstrate Squiggle in the context of CEFRIEL's activities related to Torino 2006[4]. Our aim in deploying Squiggle Ski, a service available on CEFRIEL's portal, was to help the international public of Torino 2006 in finding images of the athletes involved in the alpine skiing races.

Squiggle Ski is on-line at `http://squiggle.cefriel.it/ski`; during Torino 2006 event, it was visited by almost one thousand visitors searching for the various athletes that won a medal in the alpine-skiing races. When you open the home page, you are presented with an ordinary search box. If you try searching for "`Herminator abfahrt`" (being "Herminator" a nickname for Hermann Maier and "abfahrt" the German for downhill), you receive a plain syntactic search, and in a box on the right Squiggle Ski asks if you mean the athlete "Hermann Maier" and the discipline "downhill". If you eventually follow Squiggle Ski suggestions, all the images of Hermann Maier in a downhill race are retrieved, disregarding

[4] See also `http://www.cefriel.it/press/olimpiadi2006.html`

the language used in the initial query; an explanation box shows how Squiggle Ski expanded the query to achieve the result.

Squiggle Music – Squiggle Music is an instantiation of Squiggle framework in the music field. We noticed that both very diffuse media-players and popular sites for buying music fail to retrieve tracks when alternative wordings or translations are used (e.g. searching "rhcp" does not always retrieve the list of all Red Hot Chili Peppers tracks in the repository). Squiggle Music indexes audio files (mainly mp3 files) enriching them with information about authors, song titles and music genres. Squiggle Music is publicly available at `http://squiggle.cefriel.it/music`. Combining the *smart data* from MusicBrainz and MusicMoz[5] with a *smart machine* like QuickNamer[6] that makes use of audio fingerprints, we built an automatic semantic annotator that acts as a domain-dependent plug-in of Squiggle framework during the *Conceptual Indexing* phase. This annotator is therefore able to add to each file all its metadata (artist, song title, etc.).

From the final user's point of view, besides the usual "suggestion" of meanings, Squiggle Music is able to perform a query expansion and to present the user with other results that could be of his interest. Squiggle Music can suggest related artists when searching for a performer, songs by the same artist when looking for a song, broader and narrower styles when asking for a music genre.

6 Conclusions

In this paper, we presented Squiggle, a Semantic Web framework that eases the deployment of semantic search engines in specific applications. We enlightened how the employment of Semantic Web technologies to the development of search engines provides real benefits to end users, enabling an easier and more effective access to information; a semantic search engine, in facts, improves current syntactic engines in terms of both precision and recall, thanks to an explicit characterization of the domain at lexical and conceptual level. Semantic Web technologies show their whole potentialities in the expansion of queries to include related meanings: a semantic search engine built on Squiggle appears to be more usable, in that users are supported with semantic "suggestions", as our test-beds demonstrate at a glance.

Moreover, we designed Squiggle foreseeing possible extensions to the framework: for example, the adoption of smart machines allows the exploitation of their media-dependent capabilities and, in the meantime, the generation and aggregation of smart data.

Finally, we admit that a semantic search engine developed with Squiggle is strongly domain-dependent and cannot compete with general-purpose search engines; however, we definitely believe that such an approach provides better results, because a focused tool better meets specialized needs, helping you in finding what you're really looking for.

[5] `http://www.musicbrainz.org/` and `http://www.musicmoz.org/`

[6] `http://phonascus.sourceforge.net/`

References

1. Berners-Lee, T.: Semantic Web Road map (1998), Available on the web at http://www.w3.org/DesignIssues/Semantic.html
2. Brin, S., Page, L.: The Anatomy of a Large-Scale Hypertextual Web Search Engine. Computer Networks and ISDN Systems 30, 107–117 (1998)
3. Kiryakov, A., Popov, B., Terziev, I., Manov, D., Ognyanoff, D.: Semantic Annotation, Indexing, and Retrieval. Elsevier's Journal of Web Semantics 2(1) (2005)
4. Miles, A., Brickley, D.: SKOS Core Guide, W3C Working Draft (November 2, 2005), http://www.w3.org/TR/swbp-skos-core-guide
5. Kampman, A., van Harmelen, F., Broekstra, J.: Sesame: A generic architecture for storing and querying rdf and rdf schema. In: Horrocks, I., Hendler, J. (eds.) ISWC 2002. LNCS, vol. 2342, Springer, Heidelberg (2002)
6. Gospodnetic, O., Hatcher, E.: Lucene in action. Manning Publications (2004)