SEMANTIC WEB BROWSING

Ioannis Papadakis
Libraries and Archives Department, Ionian University, Plateia Elftherias Corfu 49100, Greece

Michalis Stefanidakis
Computer Science Department, Ionian University, Plateia Elftherias Corfu 49100, Greece

Keywords: Semantic web, ontologies, browsing, web, OWL, Ajax.

Abstract: The employment of semantic web technologies like ontologies, give the opportunity to web engineers to model the information space of web sites according to the conceptualization of the broader domain they refer to. In this paper, an innovative highly interactive semantic web browsing methodology is presented, that is applicable to a wide range of current web sites of rich and at the same time diverse content. The proposed approach is demonstrated through a prototype semantic web application based on current web technologies such as OWL and Ajax.

1 INTRODUCTION

Current semantic web applications seem not to be appealing to average web users since they transfer the complexity of semantic web technologies (i.e. OWL (www.w3.org/2004/OWL), RDF (www.w3.org/sw/RDF), ontologies, etc) to the Graphical User Interface – GUI). Consequently, average web users that are not (and should not be) accustomed to the specialized vocabulary that is employed by semantic web engineers, fail to appreciate the benefits of the provided functionality.

Having the above thoughts in mind, this paper introduces a highly interactive methodology for browsing web sites based on semantic information. Such information is described through ontologies encoded in OWL format. More specifically, depending on an underlying ontology, users are able to take control over their browsing habits by focusing on a specific part of the web site's information space.

The proposed methodology is demonstrated through the implementation of a prototype web application based on Ajax technology (Paulson, 2005). It consists of an interactive browsing interface that communicates with an ontology encoded in OWL format. The ontology's complexity is transparent to end users, since the GUI consists of distinct visualization elements (i.e. widgets) that are familiar to average web users. Reasoning with the ontology is facilitated through an open source reasoner (i.e. Pellet (www.mindswap.org /2003/pellet/)) and the standard DIG interface (Bechhofer, 2003).

The rest of the paper is structured as follows: section 2 outlines the most important issues that arise when visualizing semantic information, especially on the web. Section 3 presents the prototype semantic web application through an appropriate case study. Finally, the last section concludes this paper and points directions for future work.

2 VISUALIZING SEMANTIC INFORMATION

The initial attempts towards visualization of semantic information produced software that was delivered to end users through desktop applications. In this direction, the IsAViz tool was designed to visualize RDF graph-based structures (Pietriga, 2002). One of its key features is the fact that, in addition to showing instances, IsAViz shows connections from instances to their originating classes. Although such a tool is suitable for authoring RDF structures, it lacks usability when it is employed in browsing through metadata since users are overwhelmed with too much information and constantly crossing edges.

Papadakis I. and Stefanidakis M. (2007).
SEMANTIC WEB BROWSING.
In *Proceedings of the Third International Conference on Web Information Systems and Technologies - Web Interfaces and Applications*, pages 400-403
DOI: 10.5220/0001276204000403

A visualization technique suitable for ontologies is the "Cluster Map" developed by the dutch software vendor Aduna (www.aduna.biz). The cluster Map technique (Fluit et al, 2002) focuses on visualizing instances and their classifications according to the concepts of an underlying ontology. It is suitable for light-weight ontologies that describe a domain through a set of concepts and their relationships. Although such an approach scales relatively well for small to medium-sized ontologies, it is only suitable to ontologies featuring hierarchical relationships between their underlying concepts.

The solutions presented so far are all more or less featuring desktop standalone implementations, capable of taking the full advantage of various programming environments.

However, in order to realize Tim Berner Lee's – TBL's vision about the semantic web (Berners Lee et al, 2001), semantic applications should be readily available to users with minimum effort. Web-based solutions fulfill this requirement, since they have minimal technical prerequisites (a common web browser will do) and users are already accustomed to the overall web environment.

In this direction, the Ontolingua system (Farquhar et al, 1996) is an ontology editor and browser based on HTML widgets. Such a tool is better suited to knowledge Management – KM experts rather than common users, since it requires specific skills from the KM community. Like many others, Ontolingua employs edges and nodes to represent concepts and their relationships.

A more sophisticated approach concerning the visualization of ontologies is introduced by CNET's news.com online news site (news.com.com/The+Big+Picture/2030-12_3-5843390.html). Their ontology browsing system developed by www.liveplasma.com employs Macromedia flash technology to visualize concepts and relies on different colours to distinguish individuals that belong to different classes. However, their node-edge approach overwhelms users and the lack of zooming restricts the representation of the ontology to a minimum number of concepts and individuals.

It seems that current approaches at the field of semantic web are designed having in mind the requirements of KM experts rather than average users. In order to bring the semantic web closer to the public, this paper introduces a semantic web browsing methodology suitable to serve the average web user. As it will be described in the following sections, the interactive GUI that delivers the proposed functionality to the web browser, gradually visualizes an underlying ontology in OWL format through the employment of common HTML widgets and the most promising Ajax technology.

3 SEMANTIC WEB BROWSING APPLICATION

The proposed semantic web methodology is presented through a prototype semantic web application that visualizes ontologies in OWL format. Communication between the client-side browsing service and the server-side knowledge database is facilitated through a translation process between user-generated requests and formal semantic queries.

The design of the browsing application is defined by the following disciplines:

1. Ontological terms, while being valuable to authors of the knowledge database, should remain invisible to end users.

2. The number of widgets participating in the user interface is small. Each widget has well-defined functionality. This way, average users can familiarize themselves easier with the interface and be more effective in their search for information.

3. In order to facilitate quick navigation and fast retrieval within the underlying information space, the proposed application provides a rich set of relationships between concepts of the underlying ontology. This is done in an informal and at the same time intuitive manner, via familiar hypertext links and context menus.

4. Finally, the application is based on the ubiquitous web browser interface in order to be accessible to a great number of users in platforms with different architectures and operating systems,.

The prototype application is based on the following architecture (see fig. 1):

The KB manager and the web-driven GUI are integrated into a web-based Ajax application. Pellet is employed as an external reasoning engine, touched via the standardized DIG interface (Bechhofer, 2003). Sample test-case ontological facts are read from files in OWL format. The application's components and their corresponding functionality are explained in detail in the following paragraphs.

3.1 The Ontology KB Manager

The ontology KB manager module is a key part of the application with multiple functionalities:

1. When the browsing process is initialized by a user, the ontology KB manager loads the ontological

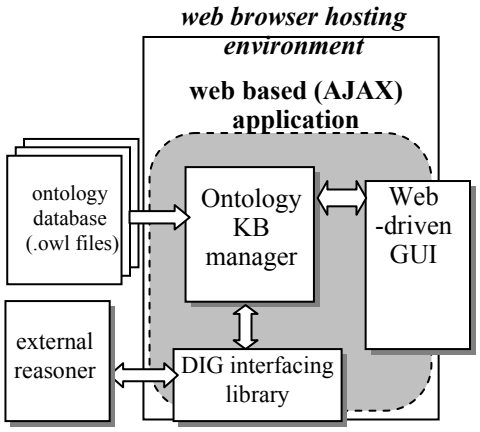

Figure 1: The architecture of the prototype application.

facts from adequate files. Such files are parsed and asserted facts are sent in an appropriate form to the external reasoner, via the DIG interfacing library.

2. The KB manager constructs a local graph of the ontology structure. Such a graph contains only part of the overall information space. This way, the web-based application is not overloaded, since the user's web browser doesn't host heavy data structures, in the case of large ontologies. Yet, due to the fact that the current DIG specification cannot convey information about the structure of an ontology back to the application, it is obligatory to keep information locally about domains and ranges of ontology relations, as well as connection between inverse relations.

3. After the initialization phase, the ontology KB manager acts as a translator of user requests to the ontology and vice-versa. In case of locally satisfied requests, results are immediately returned to the user. In any other case, the appropriate DIG query is forwarded to the external reasoner.

3.2 Semantic Browsing GUI

Interaction between users and ontologies is realized by means of four visual widgets:

a) A *ClassBox* describing a class of the ontology (see outer green box in fig. 2). The items within are the direct subclasses of this particular class. The user may navigate to a subclass repeatedly, all the way down to the bottom of the class hierarchy.

b) A *ContextMenu* (an example is shown in fig. 3), which is used to list all available relations (except of the *subclass-Of* relation), for a given class having the role of a "domain". The ContextMenu is

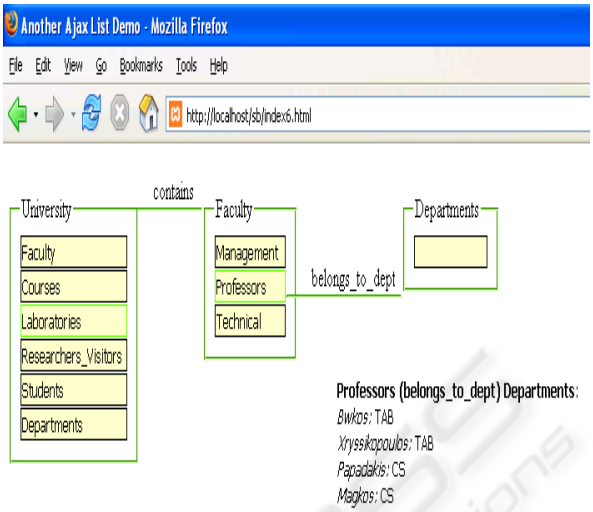

Figure 2: browsing application screenshot.

activated by right-clicking on a subclass widget of a ClassBox. The ability to navigate through the knowledge base according to the inferred relations of the ContextMenu instead of mere hierarchical subclass paths, is the true strength of the proposed interface.

The content of a ContextMenu is returned by the KB manager itself, which tracks domains and ranges of relations.

c) An interconnecting *LinkLine*, employed to denote the type of relation between two adjacent ClassBoxes. The LinkLine conveys this information via its label and positioning (see green lines in fig. 2).

Two distinct cases are possible, depending on the previous action of the user:

i) In the first case, the user navigates to the last ClassBox selecting a particular subclass widget of its parental ClassBox. The LinkLine's label (i.e. "contains" in fig. 2) depicts a subclass-Of relation. Moreover, the positioning of LinkLine at the level of both ClassBoxes' class name headers denotes a relation between these classes.

ii) In the second case (right side of fig. 2), the

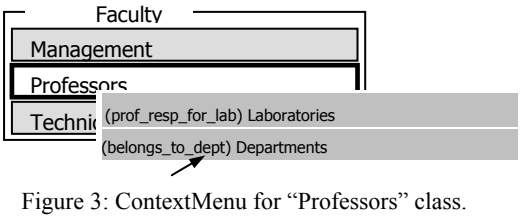

Figure 3: ContextMenu for "Professors" class.

user selects a relation from the ContextMenu (see fig. 3) of a specific subclass widget of a ClassBox. The newly created ClassBox describes a class that is

the range of the selected relation. The label on LinkLine contains the name of this particular relation, while the LinkLine height has been adjusted to indicate the subclass of the previous ClassBox, as this subclass is the domain of the depicted relation.

d) An *IndividualPane*, an informational area listing the "search results" for individuals belonging to a selected class of the ontology (see down-right part of fig. 2). These individuals are the actual targets of users' browsing actions. The term "search results" has two different meanings, in the same context as previously presented for LinkLine:

i) When a user navigates by selecting a direct subclass, the IndividualPane is connected to the last (most right) ClassBox displayed. The deriving information contains all individuals belonging to the class of this last ClassBox (i.e. current class). As this is inferred information, the KB manager requests from the external reasoner all instances of the current class in order to fill the listing area.

ii) In the case of navigation via a ContextMenu, the results displayed on the IndividualPane combine the last two ClassBoxes. These two classes are domain and range of the relation shown on the LinkLine between ClassBoxes. The IndividualPane lists now individuals related through the particular relation (down-right corner of fig. 2). For the extraction of this information the KB manager a) requests first from the external reasoner all instances (individuals) of domain class and b) in a combined request asks for each received individual about other instances (roleFillers) connected to it via the selected relation.

3.3 DIG Interfacing Library Routines

The DIG interfacing library is actually a simple wrapper based on the DIG 1.0 specification that issues asynchronous HTTP requests responsible for exchanging data in XML format between the application and the Pellet reasoner. No complex state is kept into this interface, as it is used merely for forwarding data between the ontology KB manager and the external reasoner.

4 CONCLUSIONS

In this paper, a semantic web browsing application has been presented. Specifically, the underlying resources are organized according to the conceptualization of the overall domain. Thus, resources are attached to interrelated concepts as opposed to current practice dictating that resources should be organized in classes based on the similarity of their content.

The proposed approach is demonstrated through a prototype implementation based on semantic web technologies such as ontologies and reasoners, as well as web technologies like Ajax, capable of eliminating many of today's web drawbacks. Moreover, the functionality of the approach has been further investigated through a case study that provides a web browsing interface to a sample University infrastructure. The underlying domain is conceptualized according to an OWL-DL ontology and the GUI is delivered through Ajax technology. Interaction between the ontology and the GUI is facilitated through the Pellet reasoner and the DIG interface.

Future work will be focused at extending the proposed semantic browsing methodology in a way capable of exposing editing capabilities to the underlying ontology. The ultimate goal is to provide an easy to use, multi-purpose, modular web interface between average internet users and sophisticated semantic technologies like ontologies.

REFERENCES

Pietriga, E., 2002. "IsaViz, a visual environment for browsing and authoring RDF models". In *11th International WWW Conference Developers Day*.

Farquhar, A., Fikes, R., Rice, J., 1996. *"The Ontolingua Server: a Tool for Collabo-rative Construction"*. Computer Science Department, Stanford University, Ed: BR. Gaines and MA. Musen.

Berners Lee, T., Hendler, J., Lassila, O., 2001. "The Semantic Web". In *Scientific American, Vol. 284 No 5, pp. 34–43*.

Bechhofer, S., 2003. "The DIG description logic interface: DIG/1.1." In *Proceedings of the 2003 Description Logic Workshop (DL 2003).*

Paulson, L. D., 2005. "Building rich web applications with ajax". In *IEEE Computer, Vol. 38 No. 10 pp. 14–17.*

Fluit, C., Sabou, M., van Harmelen, F., 2002. "Ontology-based Information Visu-alization:Towards Semantic Web Applications". In *Visualizing the Semantic Web*, V. Geroimenko, (Ed). Springer.