SEMANTIC WEB BROWSING

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

Thinking about information exchange today is perhaps synonymous to thinking about the web. Due to its vital role as an information highway, the web has grown in exponential rates during the last decade. Today, the web is chaotic, practically impossible to estimate its size and literally impossible to regulate. Therefore, humans can process only a tiny fraction of information available online. Consequently, we have to rely on computers to help managing and exploiting such information. Unfortunately, current web standards and practices prevent computers from understanding and interpreting the meaning (i.e. semantics) of such information in natural language form, which is how most information is represented at the web today [10]. The semantic web augments the current web by giving information a well-defined meaning, better enabling computers and humans to work in cooperation.

It is the authors’ belief that current semantic web applications are not appealing to average web users since they transfer the complexity of semantic web technologies (i.e. OWL1, RDF2, 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, relying on an underlying ontology, users are able to control their browsing habits by choosing which part of the

1 http://www.w3.org/2004/OWL
2 http://www.w3.org
information space of a web site will be presented to them.

The proposed methodology is demonstrated through the implementation of a prototype application based on Ajax technology [12]. It consists of an interactive browsing interface that communicates with an ontology encoded in OWL format. The complexity of the ontology is transparent to end users, since the GUI consists of distinct visualization elements (widgets) that are common to the web environment and therefore familiar to average web users. Reasoning with the ontology is facilitated through an open source reasoner (i.e. Pellet\(^3\)) and the standard DIG interface [11].

The rest of the paper is structured as follows: the next section tries to bring into the surface the motive for the creation of the semantic web. It is argued that such an environment borrows many concepts and technologies from the Knowledge Management community. Having the previous section in mind, section 3 outlines the most important issues that arise when visualizing semantic information, especially on the web. Section 4 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 SEMANTIC WEB VS. KNOWLEDGE MANAGEMENT

Semantics have been in the neighbourhood for quite some time now, long before Tim Berner Lee’s – TBL’s vision of the semantic web [9]. The Knowledge Management – KM community has produced considerable work aiming at instructing machines on how to comprehend the semantics of information that they possess, in order to develop software capable of managing knowledge instead of plain information.

However, despite the initial promising results, things did not evolve the way most Artificial Intelligence – AI engineers anticipated. Mapping human behaviour and wisdom proved to be particularly complex and subjective among KM researchers. Maybe the most important cause for this effect is the difficulty in setting the barriers on fundamental concepts of this domain. Each KM research group developed its own models and theories dealing with this delicate matter. Consequently, mostly monolithic systems and theories emerged, all featuring certain advantages but also drawbacks when compared against each other. The point is that the absence of collaboration between the members of this community and the inherent heterogeneity of the resulting products, prevent the wider approval of very significant findings.

On the other hand, the web evolved during the last few years in exponential rates. In contrast to the KM community, evolution on web technologies is primarily based on well-accepted standards. The diversity, bulkiness and chaotic nature of the web forced the web community to rely on standards for the foundation of this environment. No matter how ruthless the competition in this area was and still is, it was apparent from the very beginning that isolated solutions were doomed to failure. "Interoperability" and "standards" are two terms that dominate the web and lead the way towards the maturity of this information highway.

However, it was not long before it came apparent that the chaotic nature of the web required something more than interoperability to continue being useful to its users. Since the web, in contrast to other environments such as the one administered by the KM community, is primarily expected to be used by common users, it should find a way of bridging the gap between human and computer perception of the information within. The huge number and diversity of web users dictates that it is not feasible to rely on educating people about the way computers work. Instead, it seems to be more efficient to train computers to understand the way people comprehend information. In this direction, TBL, the founder of the web, envisioned the next generation of this environment.

Based on TBL’s vision, a number of semantic web standards, technologies and tools have emerged. As it will be presented in the next section, it was not long before it became apparent that visualizing semantic information is one of the main challenges that semantic web scientists have to deal with.

\(^3\) www.mindswap.org/2003/pellet/
3 VISUALIZING SEMANTIC INFORMATION

The initial attempts towards the visualization of semantic information produced software that was delivered to the users through desktop applications. Thus, the IsAViz tool was designed to visualize RDF graph-based structures [1]. One of the key features of this application 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 for browsing through metadata since users are overwhelmed with too much information and constantly crossing edges.

Similar to IsAViz, OntoViz[4] shows classes grouped with their properties and information about these properties and instances grouped with lists of their properties. These groups are connected with arcs indicating hierarchical relationships between the objects. OntoViz is an ontology visualizer employed by the popular Protege 2000 ontology editor [10]. Both OntoViz and IsAViz are based on the Graphviz engine [2].

Another software tool employed by Protege 2000 for visualizing ontologies is Jambalaya [3]. This tool is distinguished from other approaches due to the fact that it supports a zooming feature, allowing users to look into the ontology at different levels of detail.

A visualization technique suitable for ontologies is the "Cluster Map" developed by the dutch software vendor Aduna[5]. The cluster Map technique [13] focuses on visualizing instances and their classifications according to the concepts of the underlying ontology. It is suitable for light-weight ontologies that describe a domain through a set of concepts and their relationships. It employs 3-d spheres representing instances grouped in clusters connected with edges. Color plays a significant role to the Cluster Map since in distinguishes different clusters and instances. It is based on a variant of the well-known spring-embedder algorithm [4]. 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 TBL’s vision about the semantic web [9], 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, SHAKEN was introduced [5] as a web ontology browsing and editing system based on a Java-applet, HTML and server-side active lisp pages. The SHAKEN system focuses on presenting well-organized English-like concept descriptions thus hiding from users the cumbersome formal logic notation. As an editing tool, SHAKEN succeeds in providing domain experts the means to build Knowledge Bases without relying on AI specialists. However, as a browsing tool, although the presentation of both graphical and textual concepts is natural, scaling is problematic and not suitable to medium to large-scale ontologies.

Another web-based tool capable of browsing ontologies is WebOnto [6]. This approach is also based on lisp on the server-side and Java and HTML on the client-side through Java-applet technology. WebOnto employs nodes and edges to visualize the ontology as well as a list of widgets at the left part of the screen to help users selecting a subset of the overall ontology by picking a specific concept-term, thus restricting the browsing process to that term. Nevertheless, edges are crossing each other and no description is provided for the relationships between concepts.

The Ontolingua system [7] is an ontology editing and browsing system based on HTML widgets. Such a tool is better suited to 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.

The Ontosaurus [8] server has similar design principles to the Ontolingua system, since it is also based on common HTML widgets. As an editing tool for ontologies, it is better suited to KM experts due to the specialized vocabulary it employs. Moreover, the HTML frames-based logic for the
construction of the interface is outdated in current web applications.

A more sophisticated approach concerning the visualization of ontologies is introduced by CNET's news.com online news site\(^6\). Their ontology browsing system developed by liveplasma.com employs Macromedia flash technology to visualize concepts and relies on different colours to distinguish individuals that belong to different categories. 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 is safe to conclude that current approaches at the field of semantic browsing on the web are mostly designed to serve KM experts rather than common users. The employed vocabulary is unfamiliar to average users and visualization techniques do not take advantage of most recent web technologies.

Having the above thoughts in mind, the semantic web browsing methodology that is presented in this paper is based on common web technologies. As it will be described in the following sections, the GUI that delivers the proposed functionality to users, interactively visualizes the underlying ontology through the employment of common HTML widgets and the most promising Ajax technology.

4 SEMANTIC WEB BROWSING APPLICATION

The proposed semantic web methodology will be presented through a prototype semantic web application that displays ontological knowledge bases (in OWL format). Communication between the browsing service and the knowledge database is facilitated through a translation process between formal semantic queries and user-generated requests.

The main goal of the application is to maintain a balance between exposing as much semantic information as possible, without at the same time cluttering the screen with an overwhelming amount of semantic details. In this sense, the design of the browsing application is guided by the following disciplines:

1. Ontological terms, while being valuable to creators of the knowledge database, should remain invisible to end users. In the proposed application, the ontology’s schema guides the display of information transparently to end users.

2. The widgets of the user interface are kept low in number, each one with well-defined functionality. This way, common users can familiarize themselves easier with the interface and be more productive in their search for information.

3. In order to facilitate quick navigation and fast retrieval of the information being searched, the proposed application exposes enhanced connectivity between concepts of the underlying ontology. This is done in an informal and at the same time intuitive to the user manner, via familiar hypertext links and context menus.

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

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, reached via the standardized DIG interface [11]. 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.

4.1 The ontology KB manager

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

1. When user initiates the browsing process through the prototype application, the ontology KB manager loads the ontological facts from the external files indicated. Input 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. This graph contains only partial information, because it is intended to relay as many tasks as possible to the reasoner. This approach is followed in order to prevent from overloading the web-based application, and consequently the user’s web browser as well, with 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 local information 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 between user requests and the loaded ontology. If requests can be satisfied locally, results are immediately returned to the user. In any other case, the appropriate DIG query is forwarded to the external reasoner.

4.2 Semantic Browsing GUI

The interaction between users and ontologies is taking place onto the familiar web page display, by means of four visual widgets, listed in the following paragraphs. Widget examples are demonstrated through the sample ontology of a university shown in figure 2.

The four widgets used are:

a) A ClassBox (an example is shown in fig. 3), describing a class of the ontology. The contained items are the direct subclasses of this particular class. The user may navigate to a selected subclass repeatedly, all the way down to the bottom of the class hierarchy.

b) A ContextMenu (an example is shown in fig. 4), which is used to list relations (except of the subclass-Of one), to which a selected class serves as domain. The ContextMenu is activated by right-clicking on a subclass item of a ClassBox, so this is a fast way to navigate into the properties of the selected subclass. In reality, the ability to navigate through the Knowledge Base following the inferred relations of the ContextMenu instead of a mere

Figure 2: A sample ontology of a university.

Figure 3: A ClassBox example
hierarchical subclass path, 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.

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

i) In the first case, the user navigated to the last ClassBox selecting a particular subclass item of its parental ClassBox. The label on the LinkLine ("contains" – subject to internationalization) depicts a subclass-Of relation. Moreover, the positioning of LinkLine at the level of both ClassBoxes’ class name headers (“University”, “Faculty” on the left side of Fig. 5) denotes a relation between these classes.

ii) In the second case (right side of Fig. 5), the user selected a relation from the ContextMenu of a specific subclass item 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 item 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. These individuals are the actual targets of all browsing actions of the users. The term “search results” has two different meanings, in the same manner as presented previously for LinkLine.

i) When the user navigates by selecting a direct subclass, the IndividualPane is connected to the last (most right) ClassBox displayed. The information listed 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 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. 5). 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.

![Another Ajax List Demo - Mozilla Firefox](image)

Figure 4: ContextMenu for sample “Professors” class.

![Figure 5: Semantic web browsing application screenshot.](image)
4.3 DIG interfacing library routines

The DIG interfacing library is actually a simple wrapper that issues asynchronous HTTP requests exchanging data in XML format between the application and the Pellet reasoner, following the DIG 1.0 specification. 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.

5. Conclusions – Future work

The employment of semantic web technologies like ontologies to the web infrastructure gives the opportunity to model information at the back-end in a way that resembles human perception of the underlying information space. Such an approach promotes the delivery of services at the web that are human-oriented instead of technology-oriented as it is the case today, thus promoting TBL’s vision for a truly semantic web.

In this context, a semantic web browsing application has been presented based on the top-down approach. 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 bottom-up practice that dictates that resources should be organized in classes based on the similarity of their content.

The proposed approach is been 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 determining new ways of exposing the expressiveness of the underlying ontology to end users through the GUI while at the same time hiding the inherent complexity of ontologies. In this direction, the efficiency of the GUI could be further improved through the integration of new, promising web technologies such as Scalable Vector Graphics (SVG).

Moreover, ongoing work aims 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.

It is the intention of the authors to apply the proposed line of thoughts to a real-life scenario that will be formally evaluated from common web users, in order to determine the usability of the resulting semantic web application. In case of encouraging results, the inherent applicability of our approach to a wide portion of the web, will render it as a “killer-app”, capable of bringing the notion of the semantic web closer to the web community.
REFERENCES