Visualizing RDF Documents

Aris Athanassiades[1], Efstratios Kontopoulos[2] and Nick Bassiliades[2]

[1] Dept. of Business Administration, Univ. of Macedonia, GR-54006, Thessaloniki, Greece
mis0620@uom.gr

[2]Dept. of Informatics, Aristotle Univ. of Thessaloniki, GR-54124 Thessaloniki, Greece
{skontopo, nbassili}@csd.auth.gr

Abstract. The Semantic Web (SW) is an extension to the current Web, enhancing the available information with semantics. RDF, one of the most prominent standards for representing meaning in the SW, offers a data model for referring to objects and their interrelations. Managing RDF documents, however, is a task that demands experience and expert understanding. Tools have been developed that alleviate this drawback and offer an interactive graphical visualization environment. This paper studies the visualization of RDF documents, a domain that exhibits many applications. The most prominent approaches are presented and a novel graph-based visualization software application is also demonstrated.

1. Introduction

The *Semantic Web* [1] (*SW*) attempts to improve the current Web, by making Web content "*understandable*" not only to humans but to machines as well. One of the fundamental SW technologies is *XML* (*eXtensible Markup Language*) that allows the representation of structured documents via custom-defined vocabulary. However, since XML cannot semantically describe the meaning of information, *RDF* [2] (*Resource Description Framework*), an XML-based statement model, was introduced that captures the semantics of data through metadata representation.

The management of XML-based RDF documents is a task easily handled by machines that can easily process large volumes of structured data. For humans, however, the same objective is highly cumbersome and demands experience and expert understanding [3]. Software tools have been developed that alleviate this drawback, hiding the technical low-level syntactical and structural details and offering a graphical visualization interactive environment. This way, a human-user can easily create new documents or modify their structure and content.

The most substantial requirement for these software tools is the efficient visualization of RDF metadata [4]. The three most prominent RDF visualization ap-

proaches are: *display-at-once*, where the graph representing the document is displayed all at once, *navigational-centric*, where a chosen resource serves as the start-point for the rest of the graph, and *centric-graph-at-once*, a combination of the previous two. This paper studies thoroughly these approaches and demonstrates *RDFViz++*, a novel graph-based visualization software. The tool offers an alternative visualization approach that fulfills the needs unsatisfied by the available tools.

In the rest of the paper, section 2 gives some insight on RDF, followed by a section that focuses on visualizing RDF documents, presenting the three dominant visualization approaches. Section 4 presents RDFViz++, elaborating on its most distinctive features as well as its visualization algorithm. The paper is concluded with the final remarks and directions for future work.

2. RDF – A Common Information Exchange Model in the SW

RDF is a common information data exchange model describing SW resources. It consists of a number of *statements*, each being a *resource-property-value* triple: *resources* are the objects we refer to, *properties* describe attributes of resources or relations between resources and *values* can be either resources or simply literals. An example of a statement is: **<#john> <#age> <26>**, which declares that a specific person (named John) is 26 years old. Here "**#john**" is the resource (or *subject*), "**#age**" is the property (or *predicate*) and the value (or *object*) is "**26**".

In XML, RDF statements can be represented by an **rdf:Description** element. The subject is referred to in the **rdf:about** attribute, the predicate is used as a tag and the object is the tag content. Furthermore, *namespaces* provide a mechanism for resolving name clashes, when more than one document is imported. The element names are defined, using a prefix and a local name that is unique within the base URI. Additionally, external namespaces are expected to be RDF documents defining resources, used by the importing document. This allows reuse of resources, resulting in distributed collections of knowledge.

3. Visualizing RDF Documents

Since RDF is based on XML, human interaction with RDF documents becomes cumbersome, especially in rich, detailed domains with vast numbers of statements. Dedicated software utilities bring the solution: statement visualization through simple, two-dimensional shapes. A graph is usually the final result, where nodes represent resources and arrows represent predicates. Visualizing the whole document, nevertheless, is more complicated, as many resources, properties and values must be combined in one display [5]. Also, each RDF document, demands a dif-

ferent visualization approach, depending on its characteristics. A categorization of the available visualization implementations leads to three main approaches:

- *Display-at-once*: After analyzing the whole RDF document, a graph is produced that includes every single triplet. Resources are represented as rectangular or oval shapes and predicates as arrows directed from subjects towards objects. Resources involved in many statements are drawn only once; thus, multiple connections between the same resources are dealt with extended use of arrows. The greatest advantage of this approach is that it offers a complete aspect of the RDF document. However, as the document size increases, the visual result becomes unsatisfactory, due to the vast number of shapes and multiple crossings. An implementation paradigm that applies display-at-once is the well-known tool *IsaViz* [6]. An interesting feature is the zoom-in/zoom-out function in coalition with the overview map that provides the capability of better utilizing the display panel. The result is quite clear when the RDF document is not very complicated; however, complex graphs quickly become incomprehensible.
- *Navigational-centric* (or *navigational*): It is based on a chosen resource that serves as the centric node. The graph displays all the triplets, for which this node is the subject. The expansion of the rest of the graph can be interactively controlled – every object node that belongs to the already displayed statements can be chosen for further expansion. The navigational approach offers flexibility in RDF graph deployment, since it provides total control on the graph, eliminating the handicap of the display-at-once approach in visualizing "heavy" RDF documents. This is the recommended visualization approach for voluminous, complex documents and for the discovery of a specific knowledge path inside a document. However, it is not possible to have a full graph at once, a need that appears often in small and medium-sized RDF documents. An interesting implementation of this approach is the *HP Node-centric RDF Graph Visualization* [7] utility, where the navigational methodology is extended with special features like *navigation range* and *backward expansion.*
- *Centric-graph-at-once*: It is a combination of the previous two: all statements are displayed at once starting from a central node that is randomly or explicitly chosen. Arrows are designed starting from the central node; each ends at an object node, which also expands if it participates in further statements. Centric-graph-at-once has a major improvement over display-at-once: every resource that occurs in more than one RDF statements, is drawn again for every repetition. Thus, contrary to display-at-once, arrow crossings are eliminated and the result is far more legible. The method performs well on small and medium-sized RDF documents; however, as the size of the document increases, more space and processing power is needed. As a result, users can see only a part of the complete graph; the rest is displayed after interacting on the display panel. *Fentwine* [8] is an implementation based on centric-graph-at-once. It allows the user to choose a part of the graph and then zooms out the rest. This assists the user in focusing on the part he is interested in, while the application takes advantage of the rest of the screen to display as many nodes as possible.

4. RDFViz++

The most important factors in visualizing RDF are the document size and complexity. This causes different performances for different documents by the same application – every tool follows a particular non-flexible algorithm that does not adapt to document characteristics. RDFViz++ is an alternative RDF visualization approach that faces this weakness; instead of enforcing one graph style, it combines the three previous visualization techniques, preserving the advantages from each. The software offers various layouts, but even when none of them proves to be efficient enough, a random algorithmic graph layout can be executed as many times as needed, until the final result is acceptable.

The interface consists of the toolbar, the subjects list, the display panel and the status bar. Almost all functions can be executed from the toolbar at the top of the window. The central node of the graph is chosen from the subjects list on the left that contains all the subject resources of the RDF document. The rest of the screen is used for displaying the graph, except from a narrow lane at the bottom, which serves as the status bar. A snapshot of RDFViz++ is shown in Fig. 1.

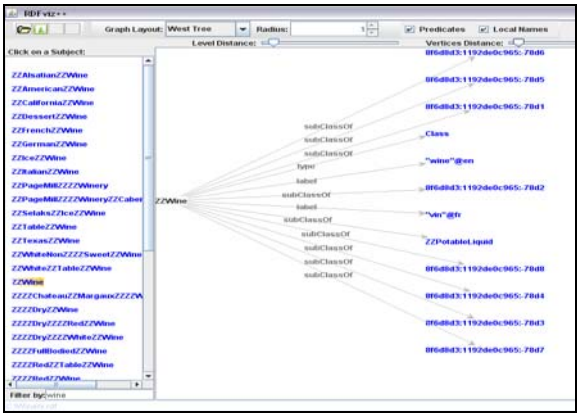

Fig. 1. RDFViz++ user interface

4.1. Visualization Algorithm

RDFViz++ features a recursion-based algorithm for graph construction. The visualization commences by choosing a centric node and continues dynamically, via two user-defined parameters: *Expansion Range* and *Node/Level Distance*. *Expansion Range* defines the expansion depth. *Level distance* is the space between different depths. *Node distance* is the space between each node that resides inside the

node group of the same level. Upon choosing the initial node, the first centric graph is constructed around it, according to the parameters above.

The central node resource is passed as a parameter to a procedure that loads all RDF statements, for which this resource is the subject. For each statement, its predicate and object are isolated and drawn as an arrow and a new node, respectively. After visualizing each statement, the system prevents re-expansion of another instance of the same resource. Nevertheless, a resource may be present more than once in a graph by following a strict constraint: it must be displayed only once as a subject and as many times as needed as an object.

The above process accepts a node and draws all adjacent nodes. If the expansion range is set to 1, then a single execution of this process gives the resulting graph. If the range is set to a greater value, then the procedure calls itself recursively. Every object that comes up from expanding the initial node becomes the procedure parameter and another call is performed. If any of the objects that arise has already been expanded as a subject before, then it is just omitted. Finally, if the range is set to 0, recursion occurs until no more objects are able to expand.

Except from the automatic graph generation, RDFViz++ also provides manual expansion via user interaction. When the initial graph is built, any visible object can be expanded, unless it has been already expanded as a subject at previous levels. Also, the RDF statements, where the resource participates as subject, must be available. If these constraints are satisfied, then the selected object is passed as a parameter to the main procedure, which is executed recursively. The number of recursive executions is equal to the number of objects that arise from the levels, which in their turn are defined by expansion range.

4.2. Graph Layouts

Document complexity does not depend only on the number of statements. One of the most significant characteristics is *concentration*, namely, the phenomenon of having only a few specific resources participate repeatedly in a vast number of statements. RDFViz++ provides a variety of graph layouts; the most appropriate can be chosen, according to the document visualization requirements. The layout can even be dynamically modified –the whole graph with the chosen layout is simply redrawn. Thus, experimentation can often lead to the best-suited configuration for each document. The available layouts are:

- *West/East/North/South Tree*: West Tree is one of the most efficient layouts, positioning the central node at the leftmost part and maintaining a left-to-right flow. In East Tree the flow is inversed. North and South Tree layouts have the same arrangement, but start deployment from the top and bottom, respectively.
- *North/West Compact*: Variations of the North and West Tree layouts that improve node placement, aiming at efficiently distributing the available space.

- *Radial Tree*: The initial centric point is the center for all levels, which are drawn as concentric circles with a greater radius than the previous levels.
- *Organic*: Uses a randomized algorithm for calculating positions.

5. Conclusions and Future Work

The paper reported on RDF document visualization, presenting the three most prominent visualization approaches. Each is suitable for specific document types, while no single methodology can handle all documents. This was the primary motivation behind RDFViz++, the RDF visualization tool presented in this work. The software adjusts to the peculiarities of the document to be visualized, offering an adequate array of available layouts and providing the possibility of choosing the most suitable approach each time. Conclusively, the software offers a more inclusive RDF visualization. Expansion range, customized level and node distances and the various graph layouts add up to a flexible interactive application.

As for future work, the software could be enhanced with various controls like zoom-in/zoom-out, inversing the flow of the arrows (from objects to subjects), overview controls etc. Furthermore, it could be enhanced with authoring capabilities; the potential of introducing, modifying or removing statements from an RDF document would transform the tool into an integrated RDF development environment. Finally, the software could also be extended with RDF Schema representation and authoring capabilities, becoming, thus, an RDF Schema *ontology editor*.

References

1. Berners-Lee, T., Hendler, J., Lassila, O.: The Semantic Web. Scientific American, 284(5), pp. 34-43 (2001)
2. Herman, I., Swick, R., Brickley, R.: Resource Description Framework (RDF). http://www.w3.org/RDF/, last accessed: 4 November 2008
3. DeFanti, T. A., Brown, M. D., McCormick, B. H.: Visualization: Expanding Scientific and Engineering Research Opportunities. IEEE Computer, 22 (8), pp. 12-25 (1989)
4. Deligiannidis, L., Kochut, K. J., Sheth, A. P.: RDF Data Exploration and Visualization. Proc. ACM First Workshop on Cyberinfrastructure: Information Management in E-Science (CIMS '07), Lisbon, Portugal, ACM, New York, pp. 39-46 (2007)
5. Frasincar, F., Telea, A., Houben, G. J.: Adapting Graph Visualization Techniques for the Visualization of RDF Data. Visualizing the Semantic Web, pp. 154-171 (2006)
6. Pietriga, E.: IsaViz: A Visual Environment for Browsing and Authoring RDF Models. Proc. 11th World Wide Web Conference (Developer's day), Hawaii, USA (2002)
7. Sayers, C.: Node-Centric RDF Graph Visualization. Technical Report HPL-2004-60, HP Laboratories, Palo Alto (2004)
8. Fallenstein, B.: Fentwine: A Navigational RDF Browser and Editor. Proc. 1st Workshop on Friend of a Friend, Social Networking and the Semantic Web, Galway (2004)