

User-Centered Incremental Data Exploration and Visualization

Leonidas Deligiannidis, Krzysztof J. Kochut, Amit P. Sheth

LSDIS Lab and Computer Science

The University of Georgia, Athens, GA 30602, USA

{ldeligia, kochut, amit}@cs.uga.edu

Abstract

In this paper we present Paged Graph Visualization (PGV), a new user-centered, semi-autonomous visualization tool for RDF data exploration, validation and examination. PGV consists of two main components: the PGV visualizer, which is the front-end of PGV, and the BRAHMS data-store, which is our high performance main-memory RDF storage system. Unlike the existing graph visualization techniques which attempt to display the entire graph, PGV has been designed for the incremental, semantics-driven exploration of very large RDF ontologies. A novel, Ferris-Wheel-like visualization technique is used to explore the so called hot spots in the graph, i.e. nodes with large numbers (hundreds) of immediate neighbors. In response to the user-controlled, semantics-driven direction of the exploration, the PGV visualizer obtains the necessary sub-graphs from BRAHMS and enables their incremental visualization without having to rearrange the previously laid out sub-graphs. PGV and BRAHMS communicate via the HTTP protocol and to provide the necessary on-line response times, BRAHMS is connected to the Web server via the Fast-CGI interface which eliminates the need of restarting the BRAHMS process at every request. Additionally, PGV has been enhanced with a speech recognizer and a speech synthesizer.

Keywords: Incremental Data Exploration, Ontology Visualization

1. Introduction

The primary objective of visualization is to present, transform, and convert data into a visual representation, so that, humans with their sharp pattern recognition skills can analyze and query the data efficiently. To increase the effectiveness of a visualization tool the tool should allow users to dynamically explore the visual representation of the data so that they can comprehend the data faster and easier. Humans can process and analyze visual representations of data remarkably faster and more effectively than doing so by reading the numerical or

textual representation of the same data [45]. As more and more data is presented to the user, the visual real-estate (e.g. a computer monitor) can become cluttered. One technique to avoid this problem is to display an overall representation of the data. However, if someone is interested in the detailed data, such as the labels of resources and relationships in a path in an RDF graph, the overall representation of the data – referred to as the *overview*, is not adequate. Even if the real-estate was infinite, laying out a large graph is computationally expensive and takes a considerable amount of time to produce. Besides the computational power needed to produce readable graphs, visualizing a vast amount of information at once may be undesirable for at least two reasons: (a) too much information may clutter the display that leads to an unorganized and disorienting visual environment, and b) the tool may become so slow that human-computer interactivity may be lost.

Interfaces in 2D have been designed for visualization results of queries in dynamic and interactive environments (e.g., InfoCrystal [42]). Even though textual representation of data is easily implemented, it fails to capture conceptual relationships. For example, semantic data allows one to richly express the movements of an individual over some area: a person P traveled from city A to city B taking bus X, in which a terrorist Q was also traveling.

PGV consists of two main sub-systems. The first subsystem, the PGV visualizer is a Graphical User Interface (GUI). The second subsystem is the backend data-store which is driven by a high performance main memory system called BRAHMS. The two subsystems communicate with each other utilizing the HTTP 1.1 [20] protocol. Even though we describe both systems in this paper, we will primarily highlight the PGV visualizer. The PGV visualizer uses an incremental algorithm to layout explored sub-graphs. We employ the tools and algorithms from GraphViz [8] [21] [17] for the graph layouts. These tools have not been designed for incremental layouts, as they tend to re-adjust the final layouts to minimize edge crossings, enforce symmetry, etc. This behavior is appealing to general purpose incremental graph layouts but it is not suitable for PGV as it will be discussed later. Instead, PGV produces simple

radial layouts at each exploration step which effectively creates the incremental layout capability.

2. Background

RDF encapsulated semantic information is normally large in size, highly interconnected, and does not follow a fixed schema [37]. Graph Visualization does not scale to large datasets [10], mainly because of the space needed to visually represent the data on a display. Thus, an exploratory technique is necessary [38] to navigate in the dataset.

Issues pertaining to semantics have been addressed in many fields such as linguistics, knowledge representation, artificial intelligence, information systems and database management. Semantic Analytics involves the application of techniques that support and exploit the semantics of information (as opposed to just syntax and structure/schematic issues [1] and statistical patterns) to enhance the existing information systems [36]. National security applications, such as aviation security and terrorist threat assessments, represent significant challenges that are being addressed by information and security informatics research [35].

As industry and academia are focusing on information retrieval over semantic metadata, it is increasingly possible to analyze such metadata to discover interesting and useful relationships. This is why visualization is becoming increasingly important in Semantic Web tools. Particularly, visualization has been used in tools that support the development of ontologies, such as semi-automatic ontology extraction tools (OntoLift [34], Text-to-Onto [32]) and ontology editors (Protégé [33], OntoLift). Such tools employ schema visualization techniques that primarily focus on the structure of the ontology, including its concepts and their relationships.

However, sometimes information about the population of the ontology (entities, instance data) is often more important than the structure of the ontology that is used to describe these instances. The Cluster Map [3] technique focuses on visualizing instances and their classifications according to the concepts of the ontology.

Some ontologies like WordNet [14] or SWETO [5] cannot be easily visualized as a graph, as they consist of a large number of nodes and edges. There are many large bioinformatics ontologies such as the Gene ontology [13], GlycO [15] and ProPreo which have several hundred classes at the schema level, and the latter two have about half a million to over six million instances, respectively.

Dynagraph [16] is a C++ platform-neutral library based on the work of GraphViz [21]. It can be used to incrementally lay out sub-graphs. One side effect of this incremental layout algorithm is that it rearranges the previously laid out nodes and edges, which could be a

desirable for a general purpose tool. Similarly, uDraw [18] provides an environment to lay out graphs and also to manipulate their location. However, the internal constraints of uDraw enforce the level of the nodes and edge crossings to be minimized leading to, sometimes, undesirable effects. For example, the user may need to place a Node at the upper left corner of the canvas. uDraw would then automatically move the Node to where ever it thinks it is best. A similar tool with the same behavior is TouchGraph [46]. TouchGraph is a spring-embedding algorithm and tool to implement customizable layouts. It is a useful tool but also deemed disruptive to users because it keeps re-adjusting the graph to a layout it determines to be best.

Jambalaya [44] is a tool for visualizing schemas and instances of ontologies. It allows users to browse the ontology detail at several levels by zooming in and out using animation to change the level of detail. The primary disadvantage of Jambalaya is that the users can get lost deep in a detailed view and as a result lose the “overview” of the visualization. The out of focus space (space containing information secondary to the user) should take less attention not to overwhelm the user but rather aid his/her focus on the point of interest by leaving the remainder space in lower level of detail [41, 40, 39].

Visualizing large ontologies has created challenges in the research on visualization. Viewing the whole graph at once is not always recommended, since the limiting factor is the visual “real estate” (the computer display).

InfoSky [25] is a system implementing a real-world metaphor, the telescope. InfoSky allows exploration of large hierarchical structured knowledge spaces, more specifically collection of documents, web pages, etc. Using the telescope mechanism, InfoSky employs a planar graphical representation of the collection; the magnification level can be adjusted to access different levels of detail.

Faceted browsing [22] is an exploration technique for large datasets. An RDF based faceted browsing implementation is shown in [9]. Faceted interfaces are better than keyword search queries and search criteria can be altered at each step of the exploration. One of the main problems with faceted browsers is the increased number of choices presented to the user at each step of the exploration.

Many [27, 26, 19] ontology based visualization tools are based on the Self-Organizing Map (SOM) [28] technique/algorithm, at least partially. WEBCOM [27] is a tool that utilizes SOM to visualize document clusters where semantically similar documents are placed in a cluster. The order of document-clustering helps in finding related documents. ET-Map [19] is also used to visualize a large number of documents and websites and uses a variation of SOM.

Spectable [12, 7] visualizes ontologies as taxonomic clusters. These clusters represent groups on instances of individual classes or multiple classes. Spectable displays class hierarchical relations while it hides any relations at the instance level. One of the main advantages of Spectable is to present a large number of instances, but with a simple ontology, to the users. Each instance is placed into a cluster based on its class membership and instances that are members of multiple classes are placed in overlapping clusters. This visualization provides a clear and intuitive depiction of the relationships between instances and classes.

In [26] a holistic “imaging” is produced of the ontology which contains a semantic layout of the ontology classes where instances and their relations are also depicted. However, this approach is not suitable to visualize instance overlap like in Spectable.

Research in visualizing multimedia enriched environments has demonstrated the need to utilize the third dimension. Examples include the visualization of maps [23, 24], tracking of activities, events, documents, etc [11], and multimedia enriched environments [29].

3. PGV Visualizer

Semantic analytics techniques for national security applications have addressed a variety of issues such as aviation safety [2], provenance and trust of data sources [30], and the document-access problem of Insider Threat [6]. Ranking, or more specifically “context-aware semantic association ranking” [4], is very useful as it finds and presents to the end-user the most relevant information of his/her search. We have developed tools and algorithms at the LSDIS laboratory to find and rank relationships of semantic information [4].

However, the user may need to explore and examine the data, and use his or her preferred criteria to find important and relevant information instead of relying on a ranking algorithm to tell him/her what is important and relevant. Sometimes, a user may simply want to explore the data without having a specific target or goal. The user may simply want to explore the data in hope of finding relationships and gain new knowledge in a different way. A user can dynamically, at every exploration step, change the exploration criteria and investigate different paths in an RDF graph representation.

The PGV visualizer consists of two subsystems. The first subsystem is the “Starter” and the second subsystem is the “Explorer”. The “Starter” is used to find and isolate the starting node/resource where the exploration will begin. After the user finds a node of interest, the “Explorer” is used to incrementally explore an RDF graph. The “Explorer” is the main subsystem used by a user to interact with PGV.

3.1. The Starter subsystem

In order for a user to explore an RDF graph, a starting point must exist; a resource where the exploration can begin. To find this starting point, the PGV Starter subsystem is used. There exist two different mechanisms for finding the starting resource. The first mechanism is the “Simple Starting Point” (SSP) where a user selects a resource from the RDF’s schema visual representation and provides a string. This is an easy and quick way of finding “a” starting point for the exploration and can be used easily by novice users. An expert user can use SSP to manually construct a complex SPARQL query to select a starting point. The second mechanism is the advanced Schema Explorer mechanism (SE). Using SE, users can build complex queries by navigating into the schema to provide specific queries.

3.1.1 Simple Starting Point mechanism. The Simple Starting Point (SSP) mechanism consists of two windows. The first window is a SPARQL query constructor. Unlike other query languages where the fields and table names of a database must be known to formulate a query, in SPARQL only relationships and class names need to be known. These relationship and class names are full URIs that people cannot easily remember. The user, however, can find all the class names from the “Schema visualizer”, which is the second window. At startup time, PGV requests the schema from the backend subsystem and visualizes the schema using the “dot” product of GraphViz. We chose the “dot” product of GraphViz for visualizing the schema because:

- it generates graphs that are directional at the overview level. These graphs can be read from left to right. The nodes are laid out in a way where all edges connecting the nodes begin on the left hand side and end on the right hand side. This enables a user to read the schema classes/nodes very easily,
- the generated graph is symmetric, wherever symmetry is applicable,
- the final graph contains minimum number of edge crossings.

For small schemas, “dot” generates the layout almost instantaneously. Moreover, laying out the schema graph is a process that only happens once, when the system first comes up. For large schemas, however, the user is advised to use the SE mechanism described below.

The user can point and click to select a class to construct a partial SPARQL query in the SPARQL window as shown in Figure 1, where the class “Person” is selected and highlighted. Then, the user types in a string in the SPARQL window, at an indicated place, to formulate a simple, yet, complete query. The result of the

query could include multiple resources that satisfy the given query. The first node in the list becomes the starting point of the exploration.

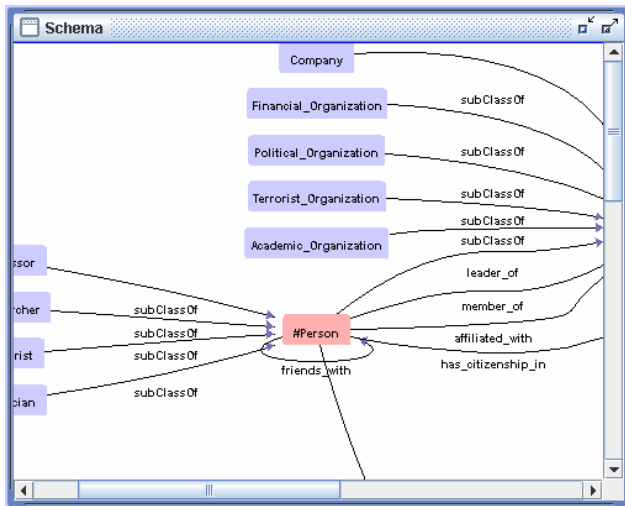


Figure 1. Schema Visualizer window with a highlighted class resource.

The SPARQL query window is an editable widget. Upon selecting a resource from the schema visualizer, a partial query appears in this window. Then the user types in the desired label string (such as the name of a town, or a person, etc) in the place indicated by the query itself as shown in Figure 2.

When the user constructs the query, the user clicks on the “Execute Query” button, shown in figure 2, and the query is sent to the BRAHMS backend system. BRAHMS then replies back with the resulting resources. The first resource returned becomes the starting point of the exploration. At this point, PGV switches to the Explorer sub-system where the user is free to start exploring the RDF data-store.

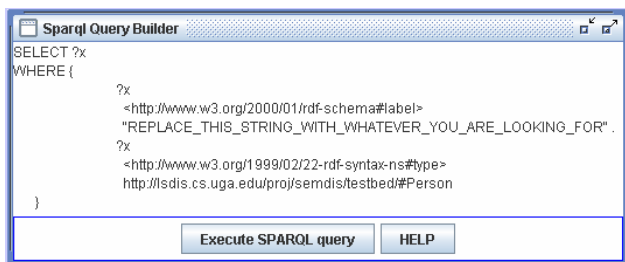


Figure 2. Starter window with a SPARQL query.

3.1.2 Sub-graph Starting Point mechanism. A user can construct a SPARQL query which returns RDF sub-graphs (Construct query). Returned sub-graphs are then

displayed and the user can select the starting node from among the shown resources.

3.1.3 Schema Explorer mechanism. Even though the SSP mechanism is easy to use, it fails when the number of classes or the relationships is large. For example, one of the ontologies we have been working on contains almost 100 classes and 50 relationships (properties). The Schema Explorer (SE) can be used for large ontologies. It allows the user to navigate in the schema and along the way to provide the criteria to narrow down the result of the query.

3.2. The Explorer subsystem

Instead of presenting to the user the entire data set, which most certainly will clutter the display with lines and shapes [43], PGV’s Explorer subsystem begins by displaying an instance that the user selects (from the Starter subsystem) along with all its direct neighbors as shown in Figure 3.

The Center node (the instance of interest) is displayed at the center of the graph in green, and its direct neighbors are shown as blue rectangles; literals are shown in white. Explored nodes are displayed in green. The edges connecting the “instance of interest” with its direct neighbors are the relationships extracted from the ontology. Their property name is displayed as a label on them (e.g. listed_author_in). The edges connecting the explored nodes are drawn in red; an example is illustrated in Figure 3. The initial layout is performed by executing the “towpi” GraphViz filter that generates radial layouts. This layout seems to be the most appropriate for PGV since we want to display an “instance of interest” along with its direct neighbors around it. The user, however, is free to move each node to any place in the canvas he/she wants. There are widgets in the application window that enable a user to “shrink” and “expand” a group of Nodes; “shrink” means that the distance between the center node and its neighbors becomes smaller, and “expand” means the opposite.

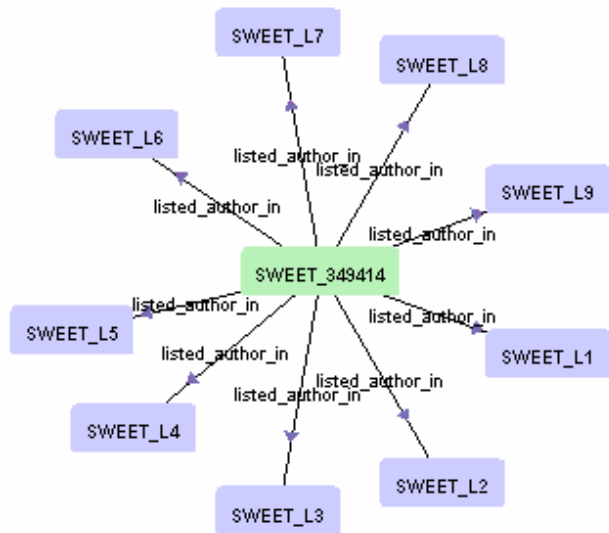


Figure 3. Center node is the instance of interest. Nodes around it are direct neighbors to it.

The user can double click on any of the neighbor nodes to explore its neighbors as shown in Figure 4. The previously laid out nodes remain at their original positions.

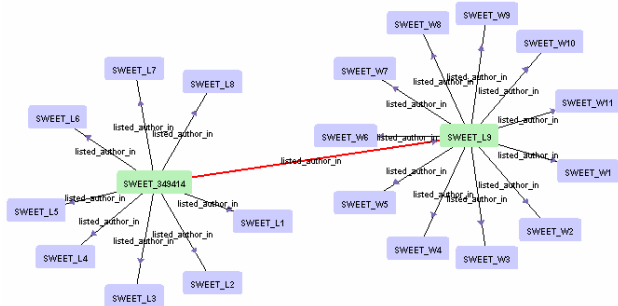


Figure 4. Exploring the neighbors of a Node.

In this example, the explored node is “SWEET_L9”. The newly explored Node reveals its neighbors and the Node itself becomes green to indicate that it is explored. The user can freely move individual Nodes (the blue nodes) or group of Nodes (the green nodes along with its direct neighbors); this depends on which node is being dragged. The edges connecting the explored nodes are highlighted in red. The user can also collapse unrelated nodes to un-clutter the display as shown in Figure 5; collapsed nodes can be expanded by double clicking on them. Nodes in blue can be hidden upon the user’s request. A double mouse click on a green (explored) node collapses or expands the node; hides and shows its unexplored neighbors respectively.

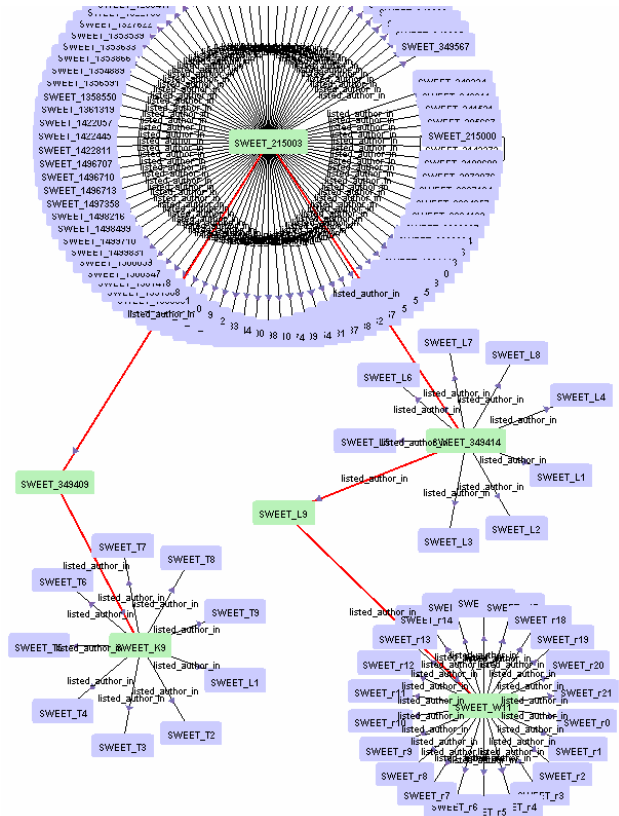


Figure 5. Data exploration with collapsed Nodes and highlighted knowledge path.

For some nodes, the number of their immediate neighbors can be very high, perhaps on the order of hundreds. We call such nodes *hot spots*. Because for such nodes the resulting radial layout may be unreadable (e.g., see Figure 5), the selection of the next node to explore may be very difficult, if not impossible. Simple zooming in is inadequate, as it makes the overall display too large. In order to handle such cases, we have created a novel technique, called the Ferris-Wheel visualization. Here, the user selects one “wheel” (the radial display of the neighbors of one node) and zooms-in sufficiently to make the labels legible. At this point only a small fragment of the wheel, say, its top is visible. However, now the user can rotate the wheel (in a Ferris-Wheel-like fashion) to quickly examine the neighbors. While one wheel is being rotated, the rest of the display remains fixed.

The user can move the mouse over an unexplored node to reveal its literals, if any, and the full URI of the node using the application’s tooltip, in order to decide if he/she wants to explore that node, as show in Figure 6.

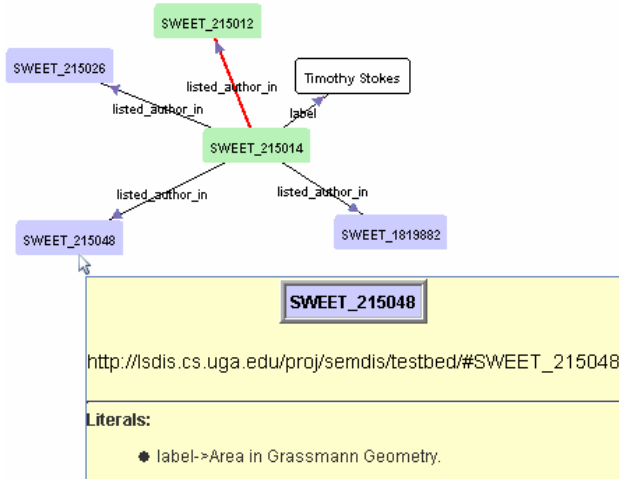


Figure 6. Tooltip information is shown for every unexplored node.

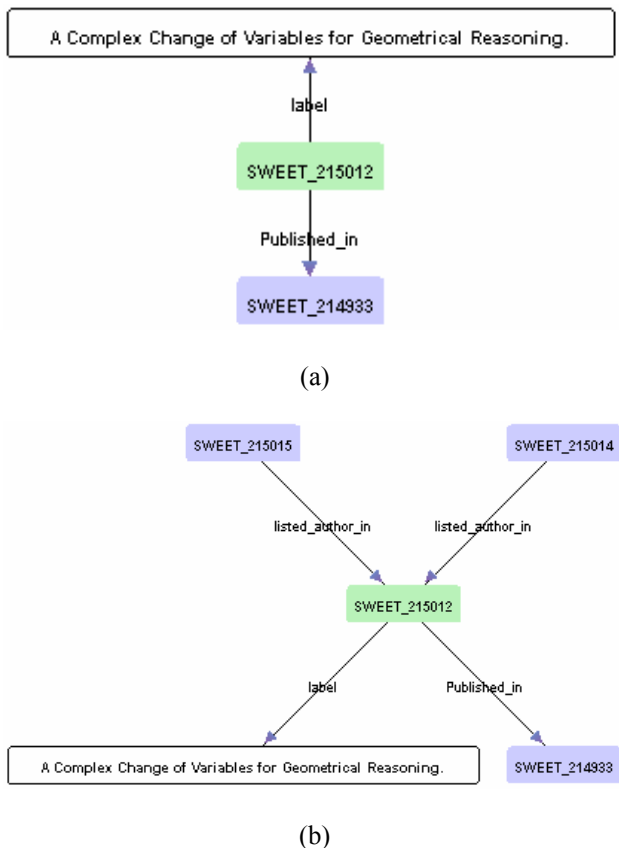


Figure 7. Two modes of exploration.

The Explorer subsystem runs in two modes: a) “Forward Exploration” (FE) and b) “Forward and Backward Exploration” (FBE). In both modes, when a node is explored, its directly connected nodes are revealed. The difference between the two modes is that

FE only displays the explored node’s neighbors that lead to them (shown in Figure 7a), while FBE includes the neighbors that lead to the explored node as well, as shown in Figure 7b.

4. PGV’s backend subsystem

The backend of PGV consists of a web server, a FastCGI program, and the BRAHMS [31] system, which is our high performance main-memory RDF data-store. BRAHMS’s high speed query performance makes it an appropriate choice for a real-time visualization system. BRAHMS is designed as a fast main-memory RDF/S storage, capable of storing, accessing and querying large ontologies. It does not use any DB backend and all data is kept in main memory. It is implemented in C++ for high performance and strict memory control.

Instead of loading the entire ontology in memory, PGV sends queries to BRAHMS (via the FastCGI interface managed by the web server) to retrieve information on demand, as shown in Figure 8. This happens when the user decides to explore a node by a double mouse click on an unexplored (blue) node.

BRAHMS’s main strength is in its speed for simple graph operations. It is achieved by its indexing scheme, usage of simple types as resource identifiers (that speeds-up all comparisons), and a read-only memory model. Results showed that such decisions resulted in high speed for graph computations.

Furthermore, division of resource types offers focused methods for graph search algorithms to concentrate only on given type (of resource or statement) and eliminates the need of filtering unwanted types during search. This caused, that during the search on instance level a user only accesses instance resources. There is no need to filter out literals or schema classes associated with instances, which makes the search simpler and more effective.

Full indexing scheme of statements (subject - predicate – object) allows for linear merge of statements in path expressions, instead of doing unsorted set intersections (which for two unsorted sets takes n^2 time). For path expression of two adjacent statements, a user can take first one sorted by "object" and a second one sorted by "subject". The result is that the resources common to both statements will be sorted in both iterators. So, finding the intersection becomes linear.

5. Architecture

PGV’s architecture is separated into two major components: the front-end (PGV visualizer) and the back-end RDF data store powered by BRAHMS. A web server is the link between the PGV visualizer and BRAHMS.

The web server, upon startup, initializes a FastCGI program that connects to BRAHMS. Queries from the PGV visualizer are sent to BRAHMS via the FastCGI program and the resulting resources and sub-graphs are sent back to the PGV visualizer. The PGV communication protocol specification can be found at <http://www.cs.uga.edu/~ldeligia/PUBLIC/PROJECTS/PGV/>.

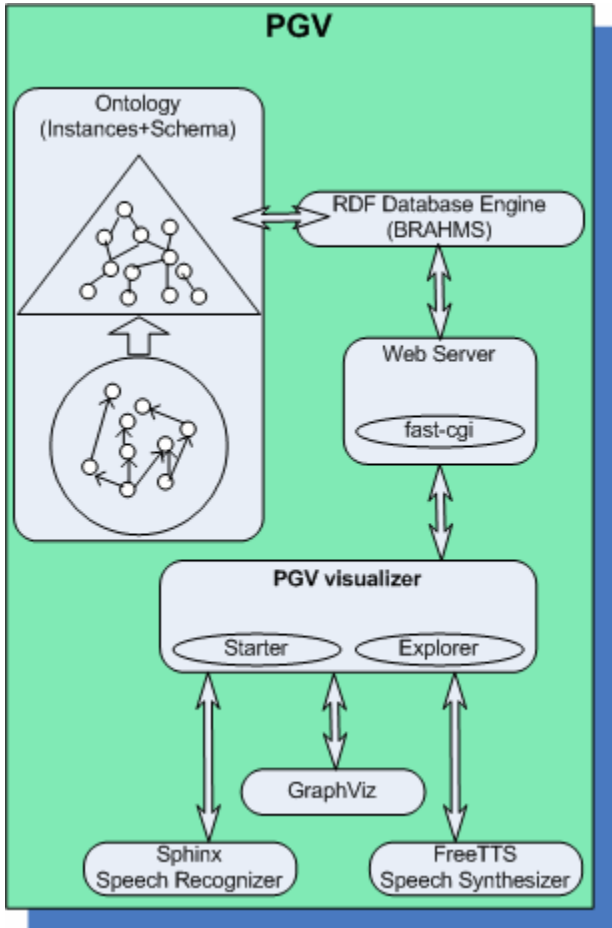


Figure 8. PGV system architecture.

The PGV visualizer is built using the Java programming language and platform. It uses the filters of GraphViz to layout the graphs.

As an example, let's assume that the user double clicks on an un-explored node. PGV extracts the full URI of the node and builds an HTTP request that looks like the following (note that the value of the "expand" variable is URL encoded – a '#' was replaced by '%23'):

```
GET /cgi/fc?expand=http://www.a.b.edu/adc/%23N1
```

It then sends the query to BRAHMS (via the web server). BRAHMS replies with a list of all connected

neighbors to the specified node (<http://www.a.b.edu/adc/#N1>). There are special characters in the HTTP protocol that are considered invalid and must be escaped. The PGV protocol also has a list of characters that are considered invalid. Thus, before transmitting and data, PGV first encodes the data (PGV encoding), followed by HTTP encode. The receiving side should decode (first HTTP and then PGV decoding), the data before using it as specified in the PGV communication protocol.

We also experimented with the Sphinx speech recognizer (<http://www.speech.cs.cmu.edu/>) and the FreeTTS speech synthesizer (<http://freetts.sourceforge.net/docs/index.php>) to enhance the application. Most of what the application is doing is expressed by the speech synthesizer. The user can also issue commands using the speech recognizer.

6. PGV communication protocol

The PGV visualizer communicates with BRAHMS via a web server utilizing the HTTP 1.1 protocol. The GET method of the HTTP is used to send requests. The PGV visualizer sends three types of requests: (a) Status, (b) Forward Exploration, and (c) Forward and Backward Exploration. The complete PGV communication protocol specification can be found at <http://www.cs.uga.edu/~ldeligia/PUBLIC/PROJECTS/PGV/net/NetProtocol.html>

6.1. Status request and response

With the Status request, the PGV visualizer checks the health of its connectivity with BRAHMS. In the GET request the variable "status", with no value, is sent. The request is shown below:

```
GET /CGI_BIN_DIR/CGI_PROGRAM?status HTTP/1.1
```

The reply is a single line that indicates a success (all is okay), or an error.

- **#status ok** indicates that everything is okay and
- **#status error error_description** indicates that something has failed. The error_description describes in more detail what the problem was.

PGV also checks for HTTP error messages in the HTTP header, in case there is a problem with the Web server itself.

6.2. FE and FBE requests

The Forward Exploration (FE) request is used to explore a node in the Forward Exploration mode. The "expand" variable is used and the full URI of the node

to be explored is assigned as a value, as shown in Figure 9.

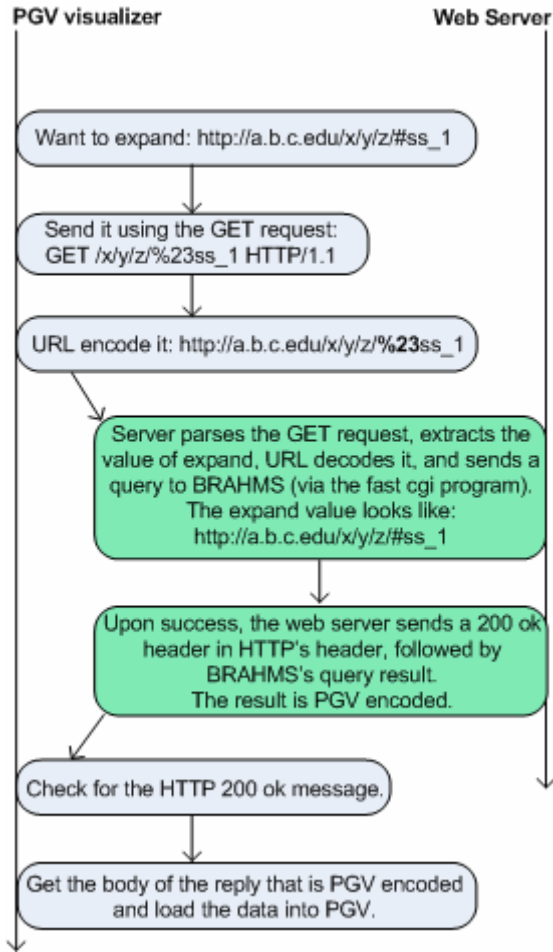


Figure 9. A request to explore a node.

An example of such a request is:

```
GET
/CGI_BIN_DIR/CGI_PROGRAM?expand=http://www.aa.bb
.edu/abc/%23SWEET_L9 HTTP/1.1
```

The value is URL encoded to comply with the HTTP 1.1 protocol. In this example the '#' was replaced with the %23.

The FBE request is similar to FE with the addition of the "allarcs" variable (with no value). An example of such a request is:

```
GET
/CGI_BIN_DIR/CGI_PROGRAM?allarcs&expand=http://w
ww.aa.bb.edu/abc/%23SWEET_L9 HTTP/1.1
```

6.3. FE and FBE responses

The reply from BRAHMS is carried in the body of an HTTP response. The reply consists of three segments

called: Header, Main Body, and Optional Body, respectively.

The Header consists of lines that begin with the '#' symbol. The very first line is a status code, same as in the Status request. Following are the prefix lines. These lines are abbreviations of URIs and have the following format:

```
#prefix prefixname1 fullname1
#prefix prefixname2 fullname2
```

The prefixname is a two character abbreviation for the fullname, which is a full URI. We needed these abbreviations to make the response easy to be read by humans. But the main reason we use the abbreviations is for GraphViz to generate small labels for the nodes and edges; GraphViz allocates enough space for each node and edge for their label to fit in. Following the prefix lines, there is a single line indicating the node of interest (the center node); this is the node the user double clicked on in the Explorer window. This line is of the form:

```
#center label
```

A label is a string of the form M*prefixname:shortname. 'M' is a single digit and it identifies the type of the resource; '1' for regular resource and '2' for a literal. The prefixname is a prefix name found in the Header section. The shortname is the short name of the resource; this is what the user sees in the PGV visualizer. An example of a label is:

```
1*p0:SWEET_2109600
```

that indicates that this is a regular resource '1', and its short name is SWEET_2109600. To get the full URI of this resource, replace p0 with the full name found in the Header section and remove the '*' and the ':' symbols which yields

```
http://lstdis.cs.uga.edu/proj/semdis/testbed/#SWE
ET_2109600
```

assuming there is a line in the Header section:

```
#prefix p0
http://lstdis.cs.uga.edu/proj/semdis/testbed/#
```

The Main Body consists of line(s) that each consists of two labels. The first label is the URI of the relationship connecting the center node to the node specified by the second label on the line.

The Optional Body consists of line(s) where each line consists of three labels. In the Forward Exploration (FE) mode, all these lines contain literals for the directly connected neighbors of the center node. The first label is a neighbor of the center node. The third label represents

the literal contained in the node represented by the first label. The second label is the relationship between the node represented by the first label and the literal represented by the third label.

In the Forward and Backward Exploration (FBE) mode, however, the optional body could also include the neighbor nodes of the center node that lead to the center node. In this case the third label is the same as the center node label.

7. Conclusion

We presented a highly interactive tool, PGV, for visualizing and exploring semantic information in very large RDF ontologies. PGV's primary goal is to present the information to the user in a simple and intuitive way. It provides an environment where users can select a small set of objects to examine dynamically in real-time, providing better contextual information. A novel, Ferris-Wheel-like visualization technique is used to explore the so called hot spots in the graph, i.e. nodes with large numbers (hundreds) of immediate neighbors. The users can select and manipulate objects using a simple point-and-click interface. We believe that PGV is a very valuable tool in data exploration and examination.

8. Acknowledgements

This research was principally supported by "SemDis: Discovering Complex Relationships in Semantic Web" funded by the National Science Foundation (NSF) grant IIS-0325464, and projects funded by ARDA. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF, ARDA or UGA.

9. References

- [1] A. Sheth, *Semantic Meta Data For Enterprise Information Integration*, *DM Review*, http://dmreview.com/article_sub.cfm?articleId=6962, Jul. 2003.
- [2] A. Sheth, B. Aleman-Meza, I.B. Arpinar, C. Halaschek, C. Ramakrishnan, C. Bertram, Y. Warke, D. Avant, F.S. Arpinar, K. Anyanwu and K. Kochut, *Semantic Association Identification and Knowledge Discovery for National Security Applications*, *Journal of Database Management*, 16 (2005), pp. 33-53.
- [3] Aduna Cluster Map Library version 2005.1, (*Integration Guide*), <http://aduna.biz/products/technology/clustermapping/docs/Aduna-Cluster-Map-2005.1-Integration-Guide.pdf>, 2005.
- [4] B. Aleman-Meza, C. Halaschek, I.B. Arpinar and A. Sheth, *Context-Aware Semantic Association Ranking*, *Proceedings of Semantic Web and DB Workshop*, Berlin, 2003, pp. 33-50.
- [5] B. Aleman Meza, C. Halaschek, A. Sheth, I. B. Arpinar and G. Sannapareddy, *SWETO: Large Scale Semantic Web Test-bed*, *International Workshop on Ontology in Action*, Banff, Canada, 2004.
- [6] Boanerges Aleman-Meza, Philip Burns, Matthew Eavenson, Devanand Palaniswami and Amit P. Sheth, *An Ontological Approach to the Document Access Problem of Insider Threat*, *IEEE International Conference on Intelligence and Security Informatics (ISI-2005)*, Atlanta, GA, 2005.
- [7] Christiaan Fluit, Marta Sabou and Frank van Harmelen, *Ontology-based Information Visualisation*, in C. Geroimenka V. Chen, ed., *Visualising the Semantic Web*, Springer Verlag London, 2002.
- [8] Emden R. Gansner and Stephen C. North, *An Open Graph Visualization System and Its Applications to Software Engineering*, *Software - Practice and Experience*, 30 (2000), pp. 1203-1233.
- [9] Eyal Oren, Renaud Delbru and Stefan Decker, *Extending faced navigation for RDF data*, *International Semantic Web Conference (ISWC)(to appear)*, Nov. 2006.
- [10] F. Frasincar, A. Telea and G.-J. Houben, *Adapting graph visualization techniques for the visualization of RDF data*, *Visualizing the Semantic Web*, 2006, pp. 154-171.
- [11] K. M. Fairchild, S. E. Poltrock and G. W. Furnas, *SemNet: Three-dimensional graphic representation of large knowledge bases*, *Cognitive Science and its Application for Human-Computer Interface*, Erlbaum, Hillsdale, NJ., 1988, pp. 201-233.
- [12] Frank van Harmelen, Jeen Broekstra, Christiaan Fluit, Herko ter Horst, Arjohn Kampman, Jos van der Meer and M. Sabou, *Ontology-based Information Visualization*, *Proc. of the workshop on Visualization of the Semantic Web (VSW'01)*, London, 2001.
- [13] Gene Ontology Home, <http://www.geneontology.org/>.
- [14] George A. Miller, *WordNet: A Lexical Database for English*, *Communications of the ACM* 38 (1995), pp. 39-41.
- [15] Glyco Ontology,
<http://lstdis.cs.uga.edu/Projects/Glycomics>.
- [16] Home page of Dynagraph, <http://dynagraph.org/>.
- [17] Home page of GraphViz, <http://www.graphviz.org/>.
- [18] Home page of uDraw, <http://www.informatik.uni-bremen.de/uDrawGraph/en/index.html>.
- [19] Hsinchun Chen, Chris Schuffels and R. Orwig, *Internet Categorization and Search: A Self-Organizing Approach*, *Journal of Visual Communication and Image Representation*, 7 (1996), pp. 88-102.
- [20] Hypertext Transfer Protocol -- HTTP/1.1 (RFC2616), <http://www.w3.org/Protocols/rfc2616/rfc2616.html>.
- [21] John Ellson, Emden R. Gansner, Eleftherios Koutsofios, Stephen C. North and G. Woodhull, *Graphviz - Open Source Graph Drawing Tools*, *Graph Drawing*, 2001, pp. 483-484.

- [22] K.-P. Yee, K. Swearingen, K. Li and M. Hearst, *Faceted metadata for image search and browsing*, In *Human-Computer Interaction (CHI'03)*, Florida, 2003.
- [23] T. Kapler, R. Harper and W. Wright, *Correlating Events with Tracked Movements in Time and Space: A GeoTime Case Study*, *Intelligence Analysis Conference*, Washington, DC., 2005.
- [24] T. Kapler and W. Wright, *GeoTime Information Visualization*, *Proc. of IEEE InfoVis*, 2004.
- [25] Keith Andrews, Wolfgang Kienreich, Verdran Sabol, Jutta Becker, Georg Droschl, Frank Kappe, Michael Granitzer, Peter Auer and K. Tochtermann, *The InforSky Visual Explorer: Exploiting Hierarchical Structure and Document Similarities*, *Information Visualization*, 1 (2002), pp. 166-181.
- [26] KeWei Tu, Miao Xiong, Lei Zhang, HaiPing Zhu, J. Zhang and Y. Yu, *Towards Imaging Large-Scale Ontologies for Quick Understanding and Analysis*, *Proceedings of the Fourth International Semantic Web Conference (ISWC2005)*, LNCS 3729/2005, Galway, Ireland, 2005, pp. 702-715.
- [27] T. Kohonen, *Self-organization of very large document collections: State of the art.*, *Proc. of the 8th International Conference on Artificial Neural Networks (ICANN98)*, 1998, pp. 65-74.
- [28] T. Kohonen, *Self Organizing Maps*, *Springer Series in Information Sciences*, Springer Espoo, Finland, 1994.
- [29] Leonidas Deligiannidis, A. P. Sheth and B. Aleman-Meza, *Semantic Analytics Visualization*, In *Proc. of the IEEE International Conference on Intelligence and Security Informatics (ISI-2006)*, San Diego, CA, USA, 2006.
- [30] Li Ding, Pranam Kolari, Tim Finin, Anupam Joshi, Y. Peng and Y. Yesha, *On Homeland Security and the Semantic Web: A Provenance and Trust Aware Inference Framework*, *The 2005 AAAI Spring Symposium on AI Technologies for Homeland Security*, Stanford University, 2005.
- [31] M. Janik and K. Kochut, *BRAHMS: A WorkBench RDF Store And High Performance Memory System for Semantic Association Discovery*, *Fourth International Semantic Web Conference ISWC 2005*, Galway, Ireland, 2005.
- [32] A. Maedche and R. Volz, *The Text-To-Onto Ontology Extraction and Maintenance System*, *ICDM-Workshop on Integrating Data Mining and Knowledge Management*, San Jose, California, USA.
- [33] Noy N. F. Sintek, M. Decker, S. Crubezy, M. Ferguson and M. A. R. W. Musen, *Creating Semantic Web Contents with Protege-2000*, *IEEE INTELLIGENT SYSTEMS Bibliographic details*, 16 (2001), pp. 60-71.
- [34] OntoLift Prototype, *WonderWeb: Ontology Infrastructure for the Semantic Web*, <http://wonderweb.semanticweb.org/deliverables/documents/D11.pdf>.
- [35] Paul B. Kantor, Gheorghe Muresan, Fred Roberts, Daniel D. Zeng, Fei-Yue Wang, Hsinchun Chen and Ralph C. Merkle (Eds.), *Intelligence and Security Informatics*, *IEEE International Conference on Intelligence and Security Informatics (ISI 2005)*, Atlanta, GA, USA, 2005.
- [36] I. Polikoff and D. Allemang, *Semantic Technology* http://www.topquadrant.com/documents/TO03_Semantic_Technology_Briefing.PDF, TopQuadrant Technology Briefing, 1 (2003).
- [37] R. Angles and C. Gutierrez, *Querying RDF data from a graph database perspective*, *2nd European Semantic Web Conference (ESWC)*, Greece, 2005, pp. 346-360.
- [38] R. W. White, B. Kules, S. M. Drucker and M. Schraefel, *Supporting exploratory search*, *Communications of the ACM*, 49 (2006).
- [39] Ramana Rao and Stuart K. Card, *The Table Lens: Merging Graphical and Symbolic Representations in an Interactive Focus+Context Visualization for Tabular Information*, *Proc. ACM Conf. Human Factors in Computing Systems*, 1994.
- [40] G. G. Robertson, S. K. Card and J. D. Mackinlay, *Information Visualization Using 3D Interactive Animation*, *Communications of the ACM*, 36 (1993), pp. 57-71.
- [41] R. Spence and M. D. Apperley, *Data Base Navigation: An Office Environment for the Professional*, *Behavior and Information Technology*, 1 (1982), pp. 43-54.
- [42] Spoerri Anselm, *InfoCrystal: A Visual Tool for Information Retrieval*, *Proceedings of the IEEE Visualization Conference*, San Jose, 1993, pp. 150-157.
- [43] Stephen G. Eick and Graham J. Wills, *Navigating Large Networks with Hierarchies*, *Proc. of IEEE Conf. Visualization*, 1993, pp. 204-210.
- [44] Storey M.A.D., Noy N.F., Musen M.A., Best C., Ferguson R.W. and Ernst N., *Jambalaya: An Interactive Environment for Exploring Ontologies*, *Proc. of the International Conference on Intelligent User Interfaces (IUI)*, 2002.
- [45] Thomas A. DeFanti, Maxine D. Brown and Bruce H. McCormick, *Visualization: Expanding Scientific and Engineering Research Opportunities*, *IEEE Computer*, 22 (1989), pp. 12 – 25
- [46] TouchGraph LLC home page, <http://www.touchgraph.com/>.