GATE II: Visualizing Semantic Web Search

Nisha Kottekat
School of Computing and Augmented Intelligence
Arizona State University
Tempe, AZ USA
Nisha.Kottekat@asu.edu

Kevin Gary
School of Computing and Augmented Intelligence
Arizona State University
Tempe, AZ USA
kgary@asu.edu*

***Abstract*—The semantic web is a mesh of information linked in a way that can be easily shared and reused to make inferences for the end user. The semantic web attempts to find and access web sites and web resources not by keywords but by descriptions of their contents and capabilities. This has been made possible by adding structure to the content of web pages and developing an environment where software agents can perform sophisticated functions for users. Semantic web customization of JMaPSS uses the Java Marker Passing Search System (JMaPSS) which applies a spreading activation search algorithm known as marker passing to significantly improve search results. This research project focuses on visualizing the semantic network and displaying marker propagation to distinctively illustrate various elements of the semantic web ontology in the form of a graph structure. The tool displays the results of marker propagation by highlighting the active nodes and the propagation path.**

Keywords*—*search, visualization, semantic web

INTRODUCTION

The exponential growth of information in the World Wide Web has resulted in an infoglut. Traditional search which is based on keyword matching burdens the user with the responsibility of creating intelligent search queries. There is an increasing need to make search intelligent, efficient, and fastidious with least human intervention. The limitation of keyword search brought forth a new paradigm, the Semantic Web [1], which can be searched through concept matching and relationship traversal. This model links various resources such as documents, images, people, or concepts semantically, moving the current web of simple relationships to a semantically rich web where meanings and new relationships can be incrementally added [2]. Figure 1 shows an example representation of a semantic web.

Knowledge in the semantic web may be encoded using the Resource Description Framework (RDF). RDF uses concepts, predicates, and relationship to provide a powerful means of expressing and inferencing relationships between resources. The Web Ontology Language (OWL) refines the Semantic Web to specify of a shared conceptualization and a vocabulary used to model a domain of interest.

JMaPSS is a platform [3] for employing semantically meaningful search through *marker-passing* [4], a parallel inference and search algorithm based on spreading activation theories of human memory organization and retrieval. In this research, we present a visualization tool for marker-passing search over the semantic web and compare the tool to prior legacy tools to demonstrate the improvements.

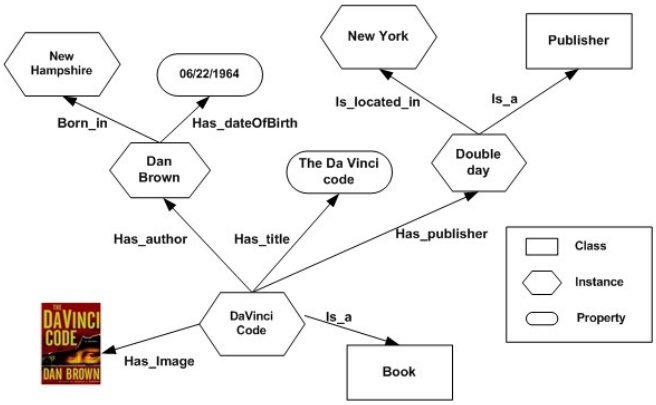


Fig 1. Representation of Semantic Web

RELATED WORK

Visualizing semantic web structures effectively remains an active area of research. A group of researchers [5][6] recently presented a comprehensive overview of the evolution of such tools, ranging from metadata explorers to graph-oriented visual tools. Several of these platforms addresses similar concerns as ours, namely displaying large semantic networks. VOWL [7] (specifically WebVOWL) visualizes semantic networks uses a forced-directed layout that has a "collapse" factor that parameterizes how many nodes to filter out when the network overwhelms the display. RDF2Graph [8] is a custom tool for biological ontologies, providing facilities to reconstruct, visualize, and query RDF structures. H-BOLD [9] (*H*igh-level visualization *O*ver *B*ig *L*inked *D*ata) handles scale by visualizing the semantic network at a "high level" and then allowing the user to drill-down to examine areas of interest, focusing more on usability and facilitating human exploration. S2SMaT [10] is a recent effort extending the usability focus by understanding the accessibility of the semantic web for ordinary users. Recent research [11] is starting to emerge to support theory-building under these efforts.

These tools show a rediscovered interest in scaling the semantic web and making it usable for a broader audience. These tools support advanced features not available in our tool, but our tool focuses on a slightly different problem – rather than explore the network from ontological and semantic query perspectives, our focus remains on keyword search that triggers spreading activation guided by the semantic information (*smart* marker-passing [12]), seeking intersections of markers to enhance search results. Therefore, a custom extension of the original GATE tool [3] was developed.

979-8-3503-2028-2/22/$31.00 ©2022 IEEE
DOI 10.1109/CSCI58124.2022.00356

SEMANTIC WEB EXTENSION OF JMAPSS

This project uses JMaPSS to represent semantic knowledge and perform parallel search. Here the semantic knowledge is exemplified in the form of ontology which is a formal representation of the shared vocabulary of a particular domain. The ontology information is described using the OWL-Lite format which is a subset of XML file format. It uses tags such as class, subclass, property, and instance to classify the different nodes. Apache Common digester [13] is used to parse the XML data and Apache Lucene [14] is used for searching and indexing. These syntax elements from Lucene are mapped to a modified JUNG [15] graph through which markers are propagated. Marker passing algorithms allows us to determine the relations between various elements that represent the semantic network.

The main interface to JMaPSS is a web application that displays the semantic network in a textual list format, listing nodes and edges. A first visual interface, GATE [3], improved the web-based interface tools used by JMaPSS by providing a graphical representation of the network thus making it easier to comprehend and traverse. However, GATE was never updated to visualize a semantic web structure and marker-passing algorithm, and had limitations in its layout capabilities that prevented view larger scale network effectively. The GATE II project, reported here, addresses these deficiencies to come up with a new, effective visualization tool.

The limitations in the current systems are twofold. The semantic web customization of JMaPSS [3] features a web-based interface that is textual rather than a graphical representation of the internal network. This depiction is hard to comprehend in terms of the nodes, its neighbors and relationship between them. This presentation limits the view of the graph at any given point of time. Graph traversal involves successive use of hyperlinks. The search result does not give an indication of the path taken by the marker or the nodes. Figure 1 shows a result of a search.

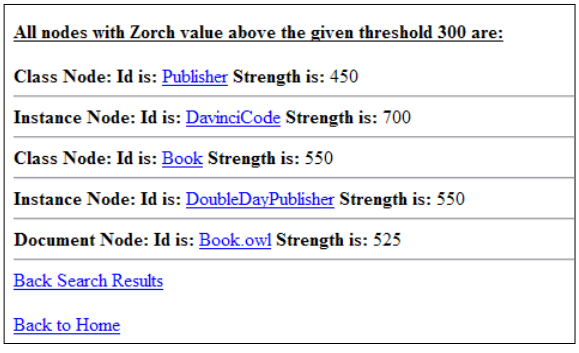


Fig 2. Search Results in the textual web interface

GATE gave a graphical visualization of network structure created by indexing of HTML text files. It represents only two types of nodes – term node and document node. In case of the GATE II, there are five types of nodes: *Class*, *Subclass*, *Property*, *Instance* and *Owl Document*. GATE currently does not depict marker-passing, does not highlight active nodes after propagation, and does not scale well for a large network. GATE II addresses all of these deficiencies.

GATE II is an extension of GATE that gives a visual display of the semantic network. The current interface is verbose and difficult to traverse. It is also difficult for the user to analyze the results of marker propagation. A graphical view of the network provides the user with a complete view of the network without the need to navigate through hyperlinks. Viewing the marker propagation results on the graph makes it easier to visualize how a marker propagates from node to node. This view can serve to differentiate the paths traversed by different heuristics.

SOFTWARE DESIGN AND ARCHITECTURE

In JMaPSS, graph elements are extracted using Lucene and used to create a JUNG data structure. The marker-passing algorithm is administered on this graph. When a user provides a search query, the query is tokenized and converted into markers. A marker is a representation of the keyword to be searched and has a unique ID, date, and a *zorch* value. A zorch value is a positive real number that denotes the activation level of the marker. The unique ID and date are useful for preventing a marker from revisiting the same node repeatedly. Once a node in the network receives the marker, it is processed and passed to the neighboring nodes. The propagation terminates once the zorch value falls below a threshold or looping occurs. The terms returned in the result are those relevant to the term the user is searching for rather than just a keyword match.

A. Architecture Overview

JMaPSS is written in Java. The relationship between node and edges are presented using JUNG. The indexed elements are extracted from the OWL files by using the Apache Common Digester. Digester provides a simple and high-level interface for mapping XML documents to Java objects. The parsed data is stored in Apache Lucene format. These indexed elements are then mapped to the JUNG graph.

GATE II uses Java Swing and Jung to visualize network structures. The three main components responsible for visualization are the *graph*, the *layout*, and the *renderer*. A graph has knowledge of the nodes in the network and the relationship between them. The layout determines the positions of the nodes in the given window. A renderer has the methods to paint nodes and edges and several parameters that control the rendering action. GATE II uses `BipartiteGaph`, `FRLayout` and the *Pluggable Renderer* classes. `VisualizationModel` is responsible for handling the graph and the layout. `VisualizationViewer` handles tracking the visualization model and the renderer, and also handles tool tip functions, pick support, and mouse listener. The zooming and panning controls are handled by `Crossover ScalingControl`. GATE provides the user the ability to filter the nodes in the graph by the strength and threshold factor.

GATE II gives the user the ability to pick and transform nodes. The pick mode allows user to select and move a single node or multiple nodes. The transform mode can be used to drag, shift-drag, control-drag to pan, rotate, and shear the graph view. The Graph Listener monitors any changes to the underlying graph and updates the view when a vertex or edge is added or removed. Node and Edge editor module handles displaying and editing the properties of the nodes or edges selected by the user.

B. Visualization

GATE II handles a graph with multiple types of nodes. The bipartite graph implementation is modified to use an undirected sparse graph. GATE II defines the colors for each of these different types of nodes in a property file. These colors are read by the program using Java reflection. This implementation gives the user the flexibility to change the colors without modifying the code. To display the weights on the edges, the weight is added as a datum or property to each edge. `EdgeWeightLabeller` is used to display the weight in the view. To change the standard size of the node elements in the graph, a new `CustomShape` function is defined. This is used to set the property of the `PluggableRenderer`.

1) Visualization of Marker-Passing

To visually annotate the graph with the active nodes and path taken by the marker, the activate function is altered and the node's datum property color is changed to the activated color. The edges are highlighted based on the state of the nodes that are connected by it. The deactivate functionality is implemented by changing the property datum to the original color.

2) Visualization of Multiple Layouts

GATE II gives the user an option to view the graph using one of four layouts: FR, Circle, ISOM or Static Layout. The layout class names are specified in the property file and read by the application using Java reflection. The layouts help in improving the organization of the nodes in a large network. We found that the FR Layout and ISOM Layout work best for most of our semantic web documents.

a) FR Layout

This layout [16] implements the Fruchterman-Reingold algorithm for node layout. The Fruchterman-Reingold Algorithm is a force-directed layout algorithm. This algorithm considers a force between any two nodes to decide the layout. It considers nodes to be steel rings and edges to be springs between them. The attractive force is analogous to the spring force and repulsive force is analogous to the electrical force. The basic idea is to minimize the energy of the system by moving the nodes and changing the forces between them. Figure 3 shows a visualization of the FR Layout.

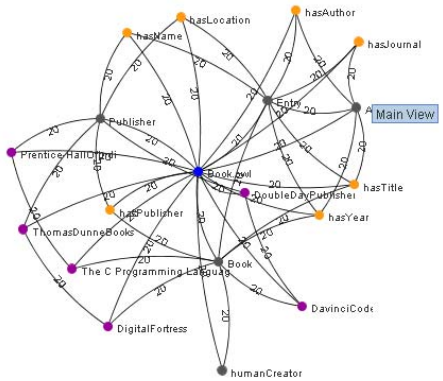


Fig 3. Visualization of graph with FR Layout

b) Circle Layout

Circle layout is a lattice-based layout algorithm where the nodes in the network are arranged in a circle. The connections between the nodes depend on the structure of the network being visualized. Figure 4 shows a visualization of a Circle Layout.

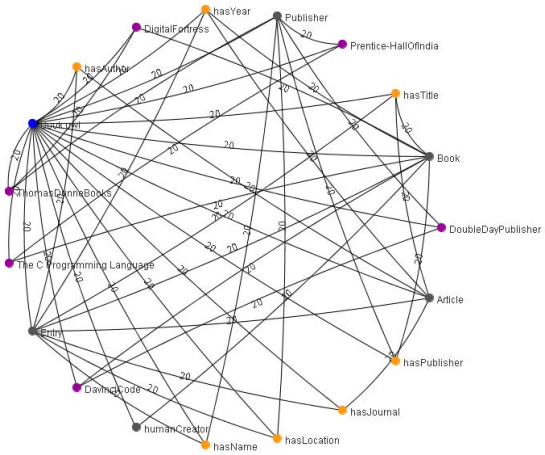


Fig 4. Visualization of graph with Circle Layout

c) ISOM Layout

ISOM layout [17] implements a self-organizing map layout algorithm based on Meyer's self-organizing graph methods. It bases its algorithm on a competitive learning strategy which is an extension of Kohonen's self-organizing maps. Figure 5 shows a visualization of ISOM Layout.

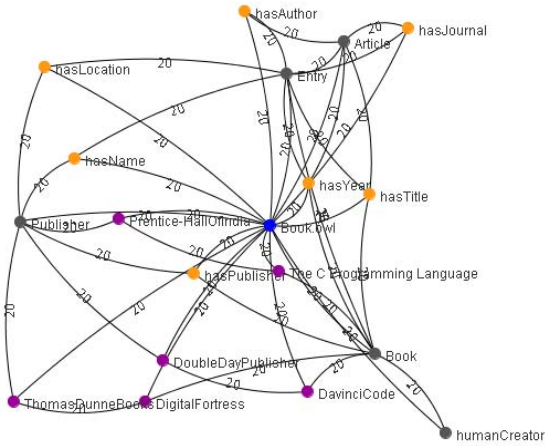


Fig 5. Visualization of graph with ISOM Layout

d) Static Layout

The algorithm used in Static Layout is specified in the JUNG API. Figure 6 shows a visualization of a Static Layout.

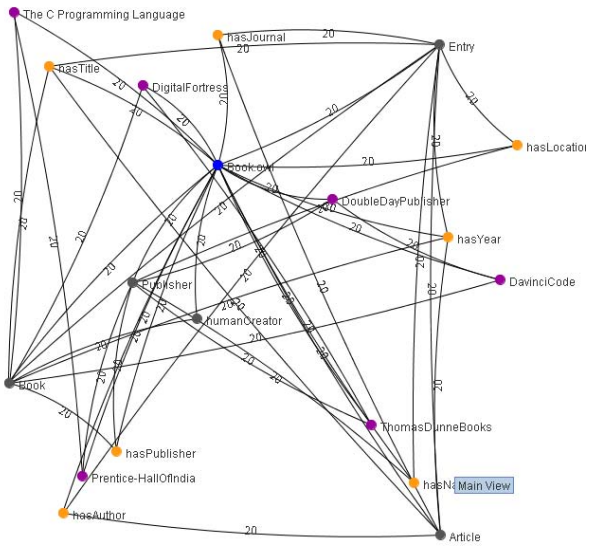


Fig 6. Visualization of graph with Static Layout

3) *Magnifying Lens Feature*

This feature was implemented by adding the magnifying lens as a Glass Pane to the viewer (Figure 7). The mouse listener keeps track of the position of the mouse pointer. It captures the area that matches the circumference of the smaller crosshair of the lens. The captured image is magnified and painted on the larger crosshair of the lens. This feature is useful for browsing larger networks by allowing for zooming in on user-desired areas as opposed to zooming the entire screen and having to scroll to a specific area of interest horizontally and/or vertically.

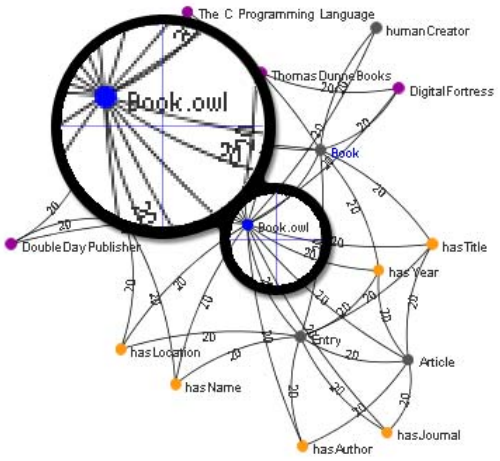


Fig 7. Visualization of magnifying lens

EXAMPLES AND ANALYSIS

The GATE II design was exercised with test cases and its functionalities were validated thoroughly. The implementation of the visualization was validated by comparing with the current system's web graph viewer. To demonstrate the tool's effectiveness the following examples are presented. The first two examples are in direct comparison to the prior text-based version of the tool shown in Figure 2.

A. *Example 1*

The goal of the example is to visualize the results of a search and emphasize the marker propagation feature of GATE II that helps analysis. Two owl files were deployed in GATE II – camera.owl that describes an analog still picture camera and digital video camera.owl that describes digital video camera. Following search queries were executed with terms that describe the digital video camera without using actual keywords like digital or video.

1) "Camera with MPEG"
2) "Camera with CCD"

This example demonstrates a search for a concept having different meanings. The result expected is digital video camera as it will be an intersection of meanings described by the query terms. Figure 9 shows the results of marker propagation after a filter is applied. The highlighted path and the nodes show the user the propagation path taken by the marker.

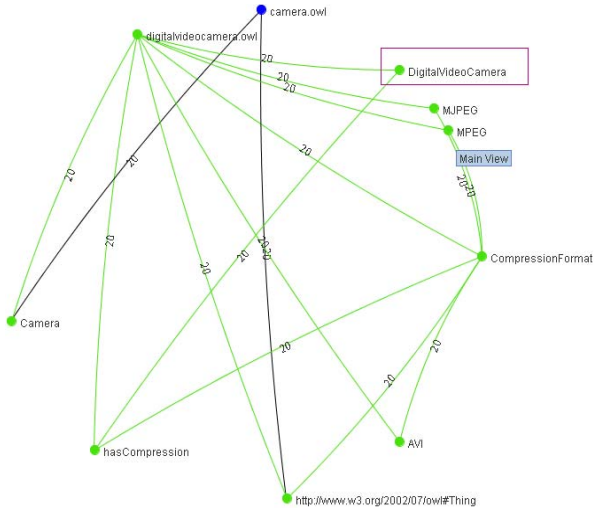


Fig 8. Graph with highlighted active nodes and propagation path

B. *Example 2*

The goal of this example is to understand the pattern of activation and propagation through visualization. Two owl files describing facial emotions were deployed in GATE II. These files have various facial emotions such as happy, sad, smile, anger, fear etc. Each file has a different description for the emotion using different terms. The following search queries were executed.

1) "furious, fuming"
2) "fright, horror"

The result for the first query is shown in Figure 10. The node *anger* has the highest strength of 700 because the node was strengthened by markers from both the terms.

The result for the second query is shown in Figure 11. The terms *fright* and *horror* result in the activation of term *fear*.

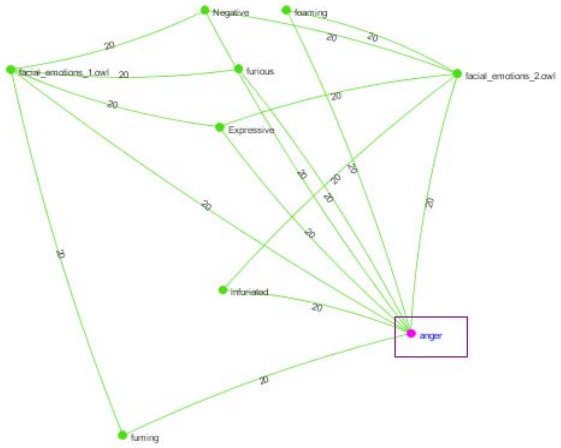


Fig 9. Result of Example 2, Query 1

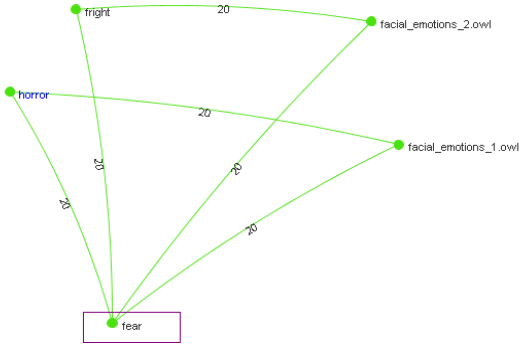
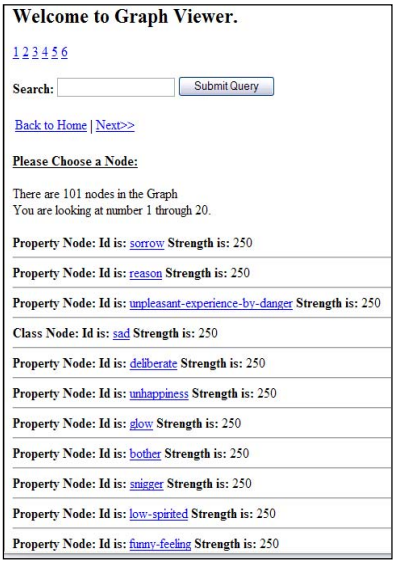


Fig 10. Result (2) of Experiment 2

C. *Example 3*

The goal of this example was to compare how well GATE II handles larger networks compared to the original text-based viewer and GATE. A somewhat larger graph (101 nodes – still not very large!) was loaded into each tool.

Figure 11 shows the output of the original text-based tool with results that spread across 6 different pages. The graph viewer does not show the connection between nodes. One has to navigate through pages and hyperlinks to explore a network.



Welcome to Graph Viewer.

1 2 3 4 5 6

Search: Submit Query

Back to Home | Next>>

Please Choose a Node:

There are 101 nodes in the Graph
You are looking at number 1 through 20.

Property Node: Id is: sorrow **Strength is:** 250

Property Node: Id is: reason **Strength is:** 250

Property Node: Id is: unpleasant-experience-by-danger **Strength is:** 250

Class Node: Id is: sad **Strength is:** 250

Property Node: Id is: deliberate **Strength is:** 250

Property Node: Id is: unhappiness **Strength is:** 250

Property Node: Id is: glow **Strength is:** 250

Property Node: Id is: bother **Strength is:** 250

Property Node: Id is: snigger **Strength is:** 250

Property Node: Id is: low-spirited **Strength is:** 250

Property Node: Id is: funny-feeling **Strength is:** 250

Fig 11. JMaPSS Graph Viewer

Figure 12 shows the output of the non-semantic web visualization tool. As there was only one layout option available the user does not have the flexibility to get a better view of the same network. As the node sizes are bigger, it is difficult to scale a larger network.

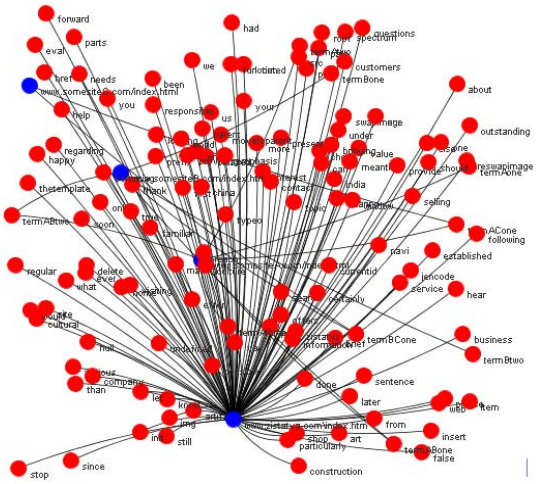


Fig 12. Larger Graph Visualization in GATE (version 1)

Figure 13 shows the output of the new semantic web visualization tool. Terms have been assigned different colors according to their types. In this example yellow nodes stand for property and grey stands for class. The layout gives a better, more scalable view of the network.

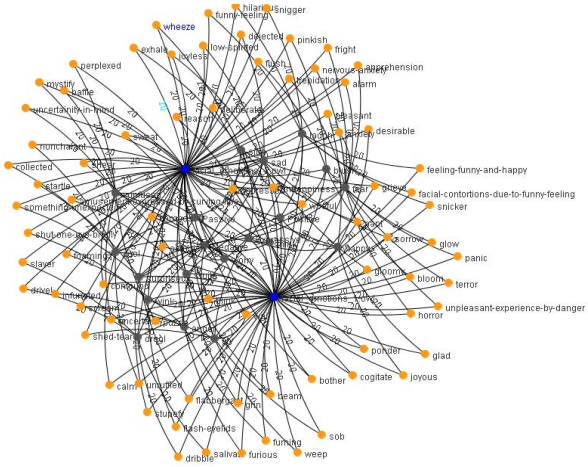


Fig 13. GATE II graph visualization

These examples demonstrate that GATE II is a robust tool to realize the Semantic Web network visually. It provides many features to help analyze the network. Some of the additional features that may be explored in the future include 3D visualization, enhanced graph coloring for search results, and adding to the available layouts.

Additional debug and visualization features can be added to help in more detailed analysis. An interesting feature that could be introduced is displaying the marker propagation results of priming of different terms in different colors. This would help differentiate the paths and intersection points. Another feature will be to have a decay by which the color of the activated nodes will fade as the strength reduces.

Currently we have explored only four layouts for visualization of graph. We could design new layouts for improving the display.

REFERENCES

[1] Berners-Lee, Tim, James Hendler, and Ora Lassila. "The semantic web." *Scientific american* 284, no. 5 (2001): 34-43.

[2] E. Miller, R. Swick (April/May 2003), An overview of W3C Semantic Web activity, Bulletin of the American Society for Information Science & Technology.

[3] Gary, K., Szabo, B., Vijayan, L., Chapman, B., Radhakrishnan, J., & Sivaraman, A. (2007, August). JMaPSS: Spreading activation search for the semantic web. In *2007 IEEE International Conference on Information Reuse and Integration* (pp. 104-109). IEEE.

[4] Gary, K., & Elgot-Drapkin, J. J. (1994, April). A flexible marker-passer for semantically weak search. In *Proceedings of the 1994 ACM symposium on Applied computing* (pp. 313-317).

[5] Po, L., Bikakis, N., Desimoni, F., & Papastefanatos, G. (2020). Linked data visualization: techniques, tools, and big data. *Synthesis Lectures on Semantic Web: Theory and Technology*, *10*(1), 1-157.

[6] Desimoni, Federico, Nikos Bikakis, Laura Po, and George Papastefanatos. "A comparative study of state-of-the-art linked data visualization tools." In *5th International Workshop on Visualization and Interaction for Ontologies and Linked Data, VOILA 2020*, vol. 2778, pp. 1-13. CEUR-WS, 2020.

[7] Lohmann, S., Negru, S., Haag, F., & Ertl, T. (2016). Visualizing ontologies with VOWL. Semantic Web, 7(4), 399-419.

[8] van Dam, J. C., Koehorst, J. J., Schaap, P. J., Martins dos Santos, V. A., & Suarez-Diez, M. (2015). RDF2Graph a tool to recover, understand and validate the ontology of an RDF resource. Journal of biomedical semantics, 6(1), 1-12.

[9] Po, L., & Malvezzi, D. (2018). High-level Visualization Over Big Linked Data. In ISWC (P&D/Industry/BlueSky).

[10] Vago, P., Sacaj, M., Sadeghi, M., Kalwar, S., Vogelsang, A., & Rossi, M. G. (2021). On the visualization of semantic-based mappings. CEUR Workshop Proceedings.

[11] Wiens, V. (2022). Visual exploration of semantic-web-based knowledge structures (Doctoral dissertation, Hannover: Institutionelles Repositorium der Leibniz Universität Hannover).

[12] Hendler, J. (1989). The design and implementation of marker-passing systems. *Connection Science*, *1*(1), 17-40.

[13] Apache Commons Digester. https://commons.apache.org/proper/commons-digester/ Last accessed September 30, 2022.

[14] O'Madadhain, J. The Java Universal Network Graph Framework (JUNG). https://jrtom.github.io/jung/. Last accessed September 30, 2022.

[15] Apache Lucene. Last accessed September 30, 2022.

[16] Fruchterman, T.M. & Rheingold, E.M. (1991) Force directed placement. Software Experience and Practice., 21.

[17] Meyer, B. (1998, August). Self-organizing graphs—a neural network perspective of graph layout. In *International Symposium on Graph Drawing* (pp. 246-262). Springer, Berlin, Heidelberg.