Visualizing Semantic Web

Meera Mudholkar
College of Engineering, Pune(COEP), India
mudholkarms09.comp-@coep.ac.in

Saloni Peswani
College of Engineering, Pune(COEP), India
peswanish09.comp-@coep.ac.in

Seema Kela
College of Engineering, Pune(COEP), India
kelask09.comp@co-ep.ac.in

Jagannath Aghav
College of Engineering, Pune(COEP), India
jva.comp@coep.ac.in

Härish Jädhäø
College of Engineering, Pune(COEP), India
jadhaohv10.comp@co-ep.ac.in

S K Hiremath
National Defence Academy, Pune, India
skhnda@rediffmail.com

ABSTRACT

Extensive volume of information spread across the web poses a major challenge for identification of relevant information. Attaching concise meaning to this information allows for effective retrieval and usage. In this paper, we propose an interactive component that extracts pertinent information from this structured data and presents it to the user through an innovative graph mechanism, namely the spring graph. Adobe Flash builder forms the base for spring graph generation, whereas deduction of desired result from semantic data is handled by Jena framework and SPARQL query language. RDF technology is used to structure information aggregated from various web sources. The current implementation reads RDF file input and visualizes it in the form of nodes of the spring graph. The solution is validated through relevant experiments that test the efficiency and proper functionality of the system. The proposed mechanism renders better quality graphs, visual output and concrete results instead of long list of documents.

Categories and Subject Descriptors

D.2.2 [**Software Engineering**]: Design Tools and Techniques—*Computer aided software engineering (CASE)*; D.2.10 [**Software Engineering**]: Design—*Methodologies, representation*; H.3.5 [**Information Storage and Retrieval**]: Online Information Services—*Web-based services*

General Terms

Theory, Algorithm and Implementation design.

Keywords

Spring graph, RDF, Jena, Sparql, Ontology

CUBE 2012, September 3-5, 2012, Pune, Maharashtra, India.

1. INTRODUCTION

Existing systems for aggregating competitive research data in the field of market analysis simply focus on the acquirement and manipulation of obtained data, with little efforts made towards presenting the data to a user. In this paper, we propose a mechanism for semantically structuring data based on the concept of semantic web and making use of available semantic web technologies. The manipulation of this structured data results in a comprehensive amount of information and discerning this in a text format is an arduous exercise for the user. To simplify this task of deciphering the information, we propose an approach to display the information in a graph visualization technique called spring graph. The system illustrates information related to company, products manufactured by the company, region, city and organization, which can be viewed by the market analyst.

A spring graph is a flex component used for graph visualization, which exhibits a graph of objects that are linked to each other using a force directed layout algorithm. The use of spring graph focuses on providing user interactivity, readability, user friendliness and output in a pleasing form. Spring graph generator will read a description file which contains the node's description and the relation between the other nodes connected to it. Each node is represented by an ellipse, whereas the connection between two nodes is represented by an arrow. The connected nodes move towards each other and this process gets repeated until the graph achieves constancy.

Adobe Flash Builder can be used for generating such spring graphs, where data can be provided in XML [1] format. It gives the graph three dimensional view. This makes the graph more reliable and readable than a general graph and focuses on good quality results. These graphs are flexible, simple, interactive and intuitive. Spring graph removes node overlapping problem which is generally found in conventional graphs.

User can perform certain operations on the results of the SPARQL queries such as searching for particular organization, product, person or any company. This result is provided to the spring graph generator. Spring graph makes use of Force based algorithms. Equilibrium Maintainer is a function of spring graph and it maintains equal distance of each node from central node. User can change repulsion

factor according to view. User can also specify how many nodes should be visible at a time. Spacing between items can also be changed according to usage. It is dynamic way of generating graphs which are more interactive.

Marketing and sales personnel, large corporate organizations and any commercial institution stand to benefit from this proposed system to satisfy the business needs of their institutes. The methodology can be used in the applications which require integration at a higher level. Industries can use it to integrate data spread across different domains. It can also be used to present company's data in a linked format and can be managed easily.

We present a general overview of the graph visualization technique called spring graph and its functionality in the introductory section. The second section focuses on a literature review of current similar systems, along with the methodology followed by our proposed implementation. A brief description of the problem that will be addressed in this paper is outlined in section 3. Section 4 presents the proposed method adopted in acquiring the solution, the detailed architecture flow and technologies used in the process. Section 5 describes an algorithm for extraction of structured data from an ontology which will be used consequently in the process of spring graph generation. Section 6 defines the actual implementation procedure in addition to the desired input and functionality provided by the proposed system. It focuses on presenting the tests that were carried out to validate the efficiency of proposed system, taking into consideration the various constraints. The results acquired from this implementation are also outlined in the section . Finally, section 6 concludes with a synopsis of the presented system and possibilities for future work.

2. RELATED WORK

A few graph visualizations have been developed for representing ontology extraction in a graphical format. But these visualizations are implemented in the conventional graph representation and no current work is being carried out in the context of spring embedded graph generation. A web based search engine based on the spring graph model provides results which are an aggregation of information obtained from varied sources like news and magazine articles, audio and video content, blog entries and many other sources. It outputs the result in a graphical representation along with charts and provides a 360-degree search and allows for varying the relevancy of a topic as per user requirement.

Another implementation produces visualization of graphs using a spring embedded algorithm that eliminates edge crossings [6]. This implementation generates three dimensional graph visualizations that provide graphs with better quality and attractiveness, although efficiency of debugging rendered by this algorithm is a point of concern.

Another paper discusses the use of graph theory for social network analysis [8]. This paper proposes that by determining the conceptual distance between people and groups, information about the type of communication in an organization can be inferred. It focuses mainly on analysis of military organizations and visualization of the relationship between people in the organization through diagrams and graphs. It makes use of graph theoretic techniques and traditional statistical approach for graph visualization. Spring embedding algorithm is used here to visualize the social network, where the spring distance correlates to the actual distance between two nodes (i.e. the link distance).The algorithm however fails to outline the actual content of communication, which plays a pivotal role in any military organization.

Yet another paper proposes an extension of the spring embedding algorithm to the three dimensional realm. This extension called the GEM-3D (Graph Embedder 3D)[9] provides graph representation at interactive speed and better display quality. It also provides graphs for real life examples accommodating hundreds of nodes and makes use of visual clues for user readability. This new algorithm can be applied to both real world graphs as well as artificial graphs to represent the topology of undirected graphs. The algorithm was also successful in presenting a visualization of the Petersen graph which was impossible to draw using the earlier spring embedding paradigm. The same implementation is applied to both directed and undirected graphs at present.

Another implementation uses spring graph to visualize the information extracted from a digital library after semantic analysis [10]. The graph representation signifies a meaningful relationship between the documents and provides an efficient way of extracting the documents. The spring embedding algorithm represents semantic relationship between the entities, with lesser spring distance representing more similarity between the nodes, and vice versa. An area of concern here is that the resources which are conceptually similar are not necessarily within proximity of each other at all times. During node stabilization of the graph visualization process, nodes can be caught up in a local minimum energy position.

This paper presents a preprocessor to enhance the performance of a conventional spring embedder, which can be used in parallel to numerous optimization and approximation techniques [11]. The preprocessor is invoked before the spring embedder and it operates in two phases. In the first phase, it maintains a user specified distance between the nodes in a graph. In the second phase, it distributes the nodes equally on a grid. The spring embedder is invoked after these two phases. This spring embedder is based on Fruchterman and Reingold algorithm which improves speed of the spring embedding process.

Current similar works provide an effective solution for extracting semantic data and utilizing it to serve as structured information for domain specific tasks. Some of the systems make use of java classes and packages for visualizing the results, which is not an altogether user friendly interface and lacks appeal of appearance. A comprehensive amount of information is aggregated from the structured ontology and representing all the information in a graph format is a tedious job, since a spring graph can accommodate limited nodes at a time. This issue needs to be addressed.

The methodology being used for visualizing the semantic web concept is dependent on the technologies. The main purpose of choosing spring graph to visualize data is that it can store data that extends existing data models. It adds new features such as defining the relationships, details of any entity, its correspondence with other entities. The graph is self explanatory and the user achieves desired output in a pleasing way. It extends the features for graph visualization which are not present in the existing data models. Spring graph makes use of RDF technology to display data in the nodes. RDF and SPARQL are the major building blocks of semantic web technology. SPARQL is used to manipulate

the RDF data. It supports Jena framework via ARQ. It is also useful for writing complex queries which helps to extend the complexity of defining data. XML is useful in defining strict hierarchy and also very reliable in parsing RDF data efficiently. XML data can also be easily converted in other format and represents any kind of data structure.

3. PROBLEM DEFINITION

Definition 1: A Spring Graph $G(V, E)$ is define as connected graph with node-set V and labeled edge-set E connecting the nodes in V. There exists label function $l : E \rightarrow R$ defined as labels from the semantic relation set R for the edges of the graph G.

Definition 2: Domain ontology $Q(C,\ I)$, where I- set of instances and C-set of concepts, there exist functions such as:

1. $h_1 : I \rightarrow P(C)$, where $P(C)$ is power set of C
2. $h_2 : I \rightarrow L$, where $L = \{l_i|$ information about i^{th} entity $\}$

Problem Definition: Given a structured data, Z, consisting of statements $\{S_i\}$ with triplet-set $(s_i,\ r_i,\ o_i)$, where $s_i,\ o_i$ belongs to entity set I and r_i belongs to relation set R, structured information visualization on Z is defined as a process that must carry out:

- Visualize and presents structured data, Z to analyst through an innovative graph mechanism, namely Spring Graph $G(V, E)$.

4. PROPOSED METHOD

The main components of system architecture are: RDF data and ontologies, reasoning engine and Spring graph generator.

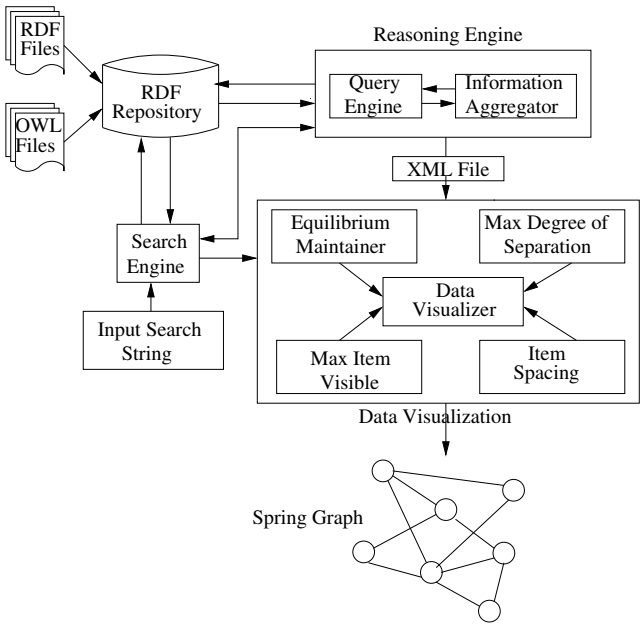

Figure 1: System Architecture

4.1 RDF Data and Ontologies

RDF, stands for Resource Description Framework, is a method used to elicit information from web pages on the worldwide web and structure it in a form understandable by a computer system. RDF represents each small piece of information in the form of a resource. A specific, domain related semantic is affixed to each resource and RDF serves as the basis for manipulating such kind of data. It displays data in triple format where every assertion is represented in following three parts: subject-predicate-object, more generally written as $P(S; O)$.

The system uses RDF file to store data and their interweaving relationships in a triplet form. A sample RDF data stored in the file is illustrated below:

```
<rdf:RDF
   xmlns:rdf="http://www.w3.org/1999/02/22−rdf syntax−ns#"
   xmlns:j.o=:http://www.owlontologies.com/swproject/GenericRelations#"
   <rdf:Description
      rdf:about="http://www.ontologies.com/swproject.owl#JCB">
      <j.0:be_powered
      rdf:resources="http://www.ontologies.com/swprojct.owl#Dieselmax"/>
   </rdf:Description>
   <rdf:Description
      rdf:about="http://www.owlontologies.com/swproject.owl#Perkins"/>
      <J.0:supplied
      rdf:resources="http://www.ontologies.com/swproject.owl#Engines"/>
   </rdf:Description>
   <rdf:Description
      rdf:about="http://www.owlontologies.com/swproject.owl#Engines"/>
      <J.0:supplied
      rdf:resources="http://www.ontologies.com/swwproject.owl#John_deere"/>
   </rdf:Description>
</rdf:RDF>
```

Figure 2: RDF Data

The literal JCB represented in the fourth line forms the subject of the RDF triple and the part preceding it is called the namespace. The namespace and literal together represent an entity. The literal Dieselmax is the object, whereas wheeled is a relationship between the subject and object and is called the predicate.

4.1.1 Ontology

Ontology [13] is description of the models and the relationships between different elements which are part of the model. It specifies the detailed relationship that holds between models. The domain of concepts and relationships get depicted well through an ontology. It is a generalization of entities in a more specific way. Ontology is a structural framework for describing a particular domain. It can also be used to reason about the entities within that domain.

4.1.2 OWL

OWL [2] is a web ontology language. It broadens the view of RDF and RDFS. Its goal is to generate an expressive and powerful language which will enhance Semantic Web technology. The syntax that OWL uses is almost similar to RDF except few changes in the vocabulary. The two main syntaxes of OWL are RDF or XML documents and abstract syntax. This emphasizes on the fact that it is a crucial part of Semantic web, it acts like an extension to RDF and RDF is compatible in parsing OWL. The abstract syntax is easy to write and read.

Proposed system uses a static ontology in the form of OWL file. Information obtained from web resources is structured in the form of company, person, organization, product

and region, since the search is constrained to business specific domain. This OWL file stores an in-depth description of all resources contained in the RDF file.

4.2 Reasoning Engine

Jena [7] is a Java Framework that supports semantic web applications and provides a number of useful tools and libraries for working on RDF data. The RDF data can be written in XML format using Jena API. The input to the system is data in the form of RDF or OWL. A file consisting of RDF [1] data or OWL [2] ontology is stored in the database in the form of subject predicate-object triple. This data is nothing but information extracted from a number of web page sources which is compiled together and organized into a structured ontology. Search is carried out on this structured data to obtain desired result. The user wants to retrieve precise information about a particular topic. The structured data stored in the RDF file is searched to get the relevant information desired by the end user. A search query is passed to the Jena and SPARQL search engine, where it is processed to yield the required result. Essentially, Jena is used as a reasoner for RDF and OWL data since it has large capacity to store RDF triplets.

The main components of a SPARQL [3] engine are the query evaluation engine and storage manager. Low level instructions are passed to the query evaluation engine, where the instructions are processed to produce the required result. File manager component of the SPARQL engine deals with data structures and memory management.

Data dictionary stores additional information of the data structure being used. Query evaluation engine has latest SPARQL specification. It provides functions like aggregate, select, update and many more, whereas storage manager is responsible for storing all the data and results of the queries.

4.3 Spring Graph Generator

Spring graph generator contains following components: Equilibrium maintainer, Maximum degree of separation, Maximum elements visible, Item spacing, and Graph visualizer.

The equilibrium maintainer maintains a constant distance between nodes in a graph. The function of this module is to check whether the distance between two nodes is the same as defined. If not, it will convert the current distance to the desired constant distance. The maximum degree of separation module determines the number of objects that will be connected to the central object. The maximum elements visible module determines the number of nodes that will be directly connected to the subnodes i.e. the objects of central node. Item spacing module ensures that the entire graph is visible in one panel as the maximum degree of separation and maximum elements values are increased. If these two values are less, the item spacing between nodes can be more. If those two values are more, the item spacing between the nodes will reduce.

5. ALGORITHM IMPLEMENTATION

The spring graph that is generated contains a large amount of nodes, with each node connected to a number of other nodes. On clicking on the link connecting any two nodes, additional information representing the relationship between these two nodes will be displayed. The textbox displaying this information will also have a link to the main news article from where this information is retrieved. Since the RDF file contains voluminous data about a large number of nodes, the major challenge is to determine which nodes will be displayed onto the spring graph. We propose an algorithm for this purpose.

Algorithm 1 Generating Spring From Structured Data

```
1: procedure RDFToSpringG(rdfData, owlOntology,
       searchEntity)
       /*Accept searchEntity from user.*/
2:     Load rdfData and owlOntology data into RDF
       triplet store.
3:     if searchEntity == Null then
4:         Goto step 8.
5:     else
6:         Goto step 9.
7:     end if
8:     searchEntity ← default_entity
       /* First element of queue forms the search entity */
9:     push searchEnity as first element on queue, Queue
10:    while Queue != Empty do
11:        subject ← Queue.pop()
12:        Execute SPARQL query on RDF store, searching
           for triplet (subject - ?predicate - ?object)
       /* Objects and corresponding relationships for */
       /* the subject are searched for in the RDF file */
13:        Execute SPAQL query on RDF store, searching
           for triplet (subject - has - ?description)
       /* Extract description related to subject from */
       /* OWL file */
14:        Store all,
           (subject, object, predicate) → tripletList
       /* store the triple on a list, to be passed for */
       /* spring graph generation */
15:        Store all,
           (subject, description) → descriptionList
       /* Store subject and its description on another */
       /* list */
16:        Queue.push(object)
       /* Store object on queue for subsequent */
       /* traversal as subject */
17:    end while
18:    CALL SpringGraph(tripletList, descriptionList)
19: end procedure
```

The proposed algorithm is simple and easy to understand. It requires limited storage in the form of a single queue and storage area for the subject-predicate-object triple. RDF data manipulation and extraction is simplified by the use of Jena and SPARQL query engine. This algorithm is efficient and trivial to implement. Search string entered by the user will be the initial subject that will be stored on the queue. The RDF file will be queried using Jena for objects related to this subject. A predefined subject is selected in case the user does not provide any input.

6. IMPLEMENTATION AND EXPERIMENTAL DESIGN

A spring graph is designed for providing information about the relationship between different entities. Adobe Flash builder provides a spring graph component class that allows building a spring graph and representing the relationship between the nodes. After clicking each link, it must dis-

play additional information about the relationship between the two entities. For the generation of spring graph, the RDF file and OWL file (containing description) are fetched. The data is extracted in triples form using SPARQL query language. The result that is obtained from the query gets stored in the form of list. The subject is pushed in the queue. The objects related to each of the subject are then extracted and added to the end of queue. This traversal is carried out until the queue becomes empty. The XML file is generated and given as input to adobe flash builder. The adobe flash builder reads the XML file to generate spring graph.

A unique feature of the system is that it allows for varying the amount of information made visible to the user. Provision is made for five sliders at the top of the panel, namely for the company, product, organization, location and person. User can select the number of these resources to be displayed at a time. As the sliders are varied by the user, the numbers of resources being written to the output XML file vary accordingly, and the corresponding nodes of spring graph are generated from this XML file.

We also present a feature of query based search, wherein the user can choose to execute his own query on the structured data, focusing on entities like company, person, organization, product and location. This constrains the search and allows the user to obtain more relevant and desired results. As an example, the user can search for companies manufacturing a particular product and situated at a specified location.

6.1 RDF and OWL File Format

The RDF file acts as input file for extracting the resources and relationships between the subject and object, whereas OWL file is the static ontology which stores the detail description about each resource. Whenever a user wants information about a particular subject, the objects and predicates related to the subject are fetched from the RDF file, whereas the description that needs to be displayed about both the subject and objects is fetched from the OWL file.

6.2 XML File Format

The processing of data in RDF and OWL files using Jena and SPARQL query language results in the generation of an XML file, which is later used for spring graph visualization. This XML file contains all the objects directly or indirectly connected to the subject being searched for by the user. It contains a detailed description of all resources enlisted next to the resource itself. The relationship between resources is also represented in this XML file, with the feature of including more than one predicates between two same resources. This XML file is read by an actionscript code in Flashbuilder for spring graph generation.

<word description="The JCB Dieselmax is a diesel-engined 'Streamliner' car designed for the purpose of breaking the land speed record for a diesel engined vehicle." id="Dieselmax"/>

The id attribute signifies the name of the actual resource and this will form the label for the nodes of spring graph. The description attribute is used to display the detail information about the corresponding resource after the cursor is pointed towards the particular node. <link a="JCB" b="Dieselmax" predicate="wheeled, be_powered, has_made, own, tracked"/>

The link tag is used for extracting information used to interconnect related resources. The attributes a and b represent the two resources, whereas the predicate attribute represents relationship between the two related resources.

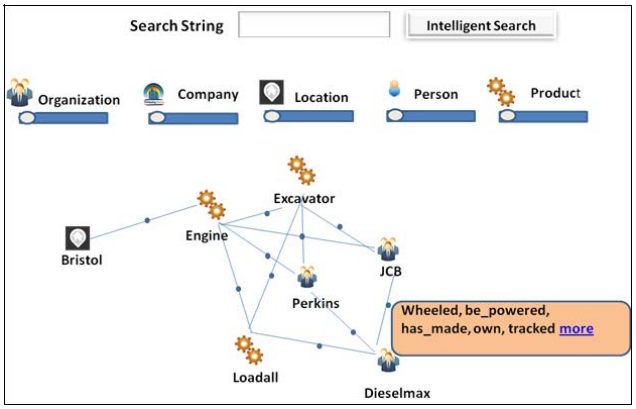

Figure 3: Spring Graph Example

6.3 Experimental Setup

Experiments were carried out to test the results obtained from the proposed system. Since spring graph is another form of graph visualization, it is restricted by the number of nodes which can be accommodated in the graph at any single time. Quite a few experiments were conducted to get an accurate estimate of the maximum amount of resources capable of being visualized by the spring graph. The structured data is stored in RDF triple store and has to be manipulated to get user defined output. This being the case, tests was carried out to deduce the amount of time required to extract the resources and corresponding relations from the massive amount of information. The efficiency of query manipulation and extraction of data rendered by the designed algorithm were also tested in comparison with conventional query manipulation techniques. Experiments were conducted to test the time complexity and efficiency of the spring graph visualization by the flex component with respect to conventional graph generation.

6.3.1 Results

The system provides an excellent tool for presenting a voluminous amount of data through the spring graph visualizer. Use of the system results in saving of time and effort on the part of the market analyst, wherein he doesn't have to scan through an enormous amount of information in order to deduce the desired result. The structured ontology leads to efficient and organized retrieval as well as manipulation of semantic data.

It became evident from the experimental design that spring graph is capable of accommodating far more number of nodes than a conventional 2D graph as it solves the problem of node overlapping. The quality of graphs generated by spring graph visualization is inarguably better, appealing and more presentable than a simple graph.

6.3.2 Discussion

Visualizing Semantic Web developments mainly concern the visualization itself and Auto Focus, as currently these are still continuously being refined. We will mainly focus on

those improvements that are relevant to information visualization and Semantic Web research.

First, we want to investigate the use of more expressive ontological structures. RDF Schema and related standards like OWL allow for much more expressive models than the hierarchical, populated classification trees visualized by the Cluster Map. High on our priority list are arbitrary binary relations between documents, since these occur frequently in the data sets we encounter. Examples are hypertext links between webpages and aggregated information objects such as emails containing attachments and zip files. Such improvements impact the query formulation user interface as well as the display of results in the Cluster Map.

In several cases we have observed that the inclusion of a time dimension in the display of information may be useful. An example is the publication date of a document. Visualizing how the population of a set of classes changes over time can greatly improve insight into how the entire document set evolves over time. Finally, we envision several AutoFocus extensions that improve its general usefulness. One example is the ability to let the user define its own taxonomy, with each class in the taxonomy being defined by a filter on the document characteristics or, when sufficient, by simply listing all the relevant documents. Similar functionality can already be found in certain mail readers, where they define virtual mail folders, and MP3 collection managers, where they define dynamic playlists. In our context, this concept can be extended to arbitrary document sets. Additionally, since AutoFocus and Metadata Server share the same Sesame-based back-end, this can provide a basis towards a P2P version of AutoFocus. Only the typical P2P communication layer has to be provided, a query answering mechanism is already in place. These two extensions may reinforce each other, as the possibility to share (populated) taxonomies may make their creation and maintenance well worth the effort.

7. CONCLUSIONS

The proposed implementation of the spring graph component forms a robust foundation for information retrieval and visualization. In this implementation, we have presented the extraction of meaningful, relevant results from a well structured ontology file and also demonstrated a solution for efficient visualization using spring graph generation. Generating spring graph through Adobe Flash builder helps in presenting information to the user in an effective, interactive form wherein precise information is displayed in the form of nodes and relationship between resources is represented by the interconnecting links. Provision is also made to view the original web source from where the information is extracted. In future, we plan to determine the similarity between resources and accordingly vary the length of links, with more similarity resulting in shorter distance between nodes and vice versa. Also similar resources will be placed together in a cluster, with each cluster having a unique color for efficient identification.

8. REFERENCES

[1] S. Decker, S. Melnik, F. V. Harmelen, D. Fensel, M. Klein, M. Erdmann, *The Semantic Web:The Roles Of XML and RDF*, 2000.

[2] Milea V., Frasincar F.,Kaymak U. *A temporal Web Ontology Language.*

[3] J. Huang, D. J. Abadi, K. Ren, *Scalable SPARQL Querying of Large RDF Graphs.*

[4] F. Bajramovic, A. Tauber,R. Wozelka and W. Ferdinand,*A Taxonomy of Force-Directed Placement Techniques.*

[5] P. Cimiano, J. Volker. *Text2Onto, A Framework for Ontology Learning and Data-driven Change Discovery.*

[6] Weimin Wu, Yongfeng Cao, Quing Su, Yonghe Zhang, Guangdong.*Visualization of Graph based on the Three-dimensional Spring Model.*

[7] J. J. Carroll, I. Dickinson, Chris Dollin, D. Reynolds, A. Seaborne, K. Wilkinson. *Jena: Implementing the Semantic Web Recommendations*, 2003.

[8] A. Dekker. *Visualization Of Social Networks Using CAVALIER*,2006.

[9] Ingo Brub and Arne Frick. *Fast Interactive 3-D Graph Visualization*, Germany.

[10] J. Zhang, C. Hill, J. Mostafa, H. Tripathy. *Information Retrieval by Semantic Analysis and Visualization of the Concept Space of D-Lib Magazine*, 2010.

[11] Paul Mutton, Peter Rodgers.*Demonstration of a Preprocessor for the Spring Embedder*

[12] Jennifer Golbeck and Paul Mutton,*Spring-Embedded Graphs for Semantic Visualization.*

[13] Nicola Guarino and Pierdaniele Giaretta. *Ontologies and Knowledge Base.*

[14] Claus Pahl and Edmond Holohan, *Applications of Semantic Web Technology to Support Learning Content Development*, Dublin City University, Dublin, 2009.

[15] Java, Jruby, Scala and Clojure Edition. *Practical Semantic Web and Linked Data Application*,2011.

[16] Andrew Mathews, Architect and developer, Freelance. *Undersranding SPARQL (Create journaling micro-blogs with the semantic web)*, 2008.

[17] Olaf Hartig, Christian Bizer and J. C. Freytag. *Executing SPARQL Queries over the Web of Linked Data*, Humboldt University, Berlin.

[18] *"Resource Description Framework (RDF)"*, http://www.w3.org/RDF.

[19] *"SPARQL"*, http://www.w3.org/TR/rdf-sparql-query.

[20] *"Web Ontology Language (OWL)"*, http://www.w3.org/TR/owl-features.