An Approach for Comparative Research Between Ontology Building & Learning Tools for Information Extraction & Retrieval

Dr Suresh Jain¹, C.S.Bhatia², Dharmendra Gupta³, Sumit Jain⁴ & Bharat Pahadiya⁵

Abstract—Information available on the web is huge & it covers diversified fields. Nowadays most of search engines use essentially keyword based search techniques. We simply specify a set of keywords or query as a request and a reference we get a list of pages, ranked based on similarity of query. Currently searching web face with one problem that many times outcome is not satisfactory because of irrelevance of the information. Searching the exact information from such a huge repository of unstructured web data is still main area of research interest. One solution to achieve this is Semantic Web. Ontology is an effective concept commonly used for the Semantic Web. Ontology is “an explicit specification of a conceptualization”. There are two main pillars of semantic Web one is Problem Solving Methods & another is Ontology. Ontology building is a tedious job and a time consuming task for user. The quality of ontology plays an important role in information retrieval application. This paper deals with features & familiarity with different Ontology building & learning tools. After all the preliminary knowledge about all tools & software we have made research about specific features & services provided by some tools & identified the optimum tool in all respect for particularly for our further research project.

Key words— Protégé 3.4, IsaViz, Apollo, SWOOP, Ontoedit

I. INTRODUCTIONS

The Semantic Web [1] is an extension of the current web in which information is given well-defined meaning, better enabling computers and people to work in cooperation. The Semantic Web will bring structure to the meaningful content of Web pages, creating an environment where software agents roaming from page to page can readily carry out sophisticated tasks for users [1]. It contributes several mechanisms that can be used to classify information and characterize its context for intelligently retrieving information on web. This is mainly done using knowledge representation languages that create explicitly domain conceptualizations, known as ontology [2], [3]. This conceptualization consists of a set of concepts, their definition and the relationships between them. Ontology is recognized as the backbone or main pillar to support various types of information management including information retrieval, storage, and sharing on web. The building & learning of ontology demands the use of various software tools [4]. A range of open-source and commercial tools are available which assist in the development of various ontology Called Ontology Editors. These tools can be applied to all stages of the ontology life cycle including the creation implementation, and maintenance of ontology. This paper

Comparative research between different ontology editor tools for the ontology construction & we have selected an ontology editor for our further research projects.

Ontology is becoming the main pillar of the Semantic Web. Ontology aim at capturing domain knowledge in a generic way and provide a commonly agreed understanding of a domain. They are shared conceptualizations of a domain and they include the representation of these conceptualizations [5]. They are used to facilitate efficient exchange of information among people, now used for communication among software agents. Ontology is independent from the applications that use them. This leads to easier software and knowledge maintenance, and contributes to the semantic interoperability between applications. Today a variety of developing environments exist for building ontology like Protégé 3.4, IsaViz, Apollo, and SWOOP[6], [7], [8], [9]. Protégé 3.4 [6] is a knowledge based ontology editor providing graphical user interface. It is selected because it provides better flexibility for meta-modeling, enables the construction of domain ontology; customize data entry forms to enter data. It is typically targeted at the knowledge engineering and conceptual modeling without knowing or thinking about syntax of output language. IsaViz [7] is a visual environment for browsing and authoring RDF models as graphs. This tool is offered by W3C Consortium. IsaViz [41] was developed by Emmanuel Pietriga. IsaViz imports RDF/XML and N-Triples, and exports RDF/XML, N-Triples, Portable Network Graphics (PNG) and Scalable Vector Graphics (SVG). Therefore, it is possible to import ontology to other editors, for instance, Protégé or OilEd. The IsaViz environment is composed of four main windows: the IsaViz RDF Editor window, the Graph window, the Definition window and the Attribute window. Apollo [8] is a user-friendly knowledge modeling application. The modeling is based around the basic primitives, such as classes, instances, functions, relations.
II Ontology Building

This section presents, in direct chronological order, the most well known approaches for building ontology [10] from scratch, as well as reusing ontology that are stored in ontology libraries. There is no one correct methodology for developing ontology. Developing ontology is usually an iterative process. We can start with a rough first pass at the ontology and then revise and refine the evolving ontology. Ontology is a model of a real domain in the world and the concepts in the ontology must reflect this reality. After defining an initial version of the ontology, we can evaluate and debug it by using it in applications or problem-solving methods or by discussing it with experts in the field. As a result, we will almost certainly need to revise the initial ontology. This process of iterative design will likely continue through the entire lifecycle of the ontology.

Developing Ontology may include:
- Selection of Domain and Scope
- Consider Reuse
- Find out Important Terms
- Defining Classes and Class Hierarchy
- Defining Properties of Classes and Constraints
- Create Instances of classes

To construct an ontology one must have an ontology specification language, of which there are several to choose. Among many ontology languages, the Web Ontology Language (OWL) is the widely accepted as standard for representing and sharing knowledge in the Semantic Web context.[11] OWL is based on Resource Description Framework (RDF) and the DARPA Agent Markup Language (DAML) [12].OWL would use the RDF meaning of classes and properties (rdfs: Class, rdfs: subclass Of, etc.) and would add some very powerful modeling primitives to extend the expressiveness. OWL also provides an owl: imports construct which syntactically includes the complete referenced ontology into the importing ontology [13],[14]. This construct does not allow partial reuse but can only handle complete ontologies. When starting out on an ontology project, the first and reasonable reaction is to find a suitable ontology software editor. These tools can help acquire, organize, and visualize the domain knowledge before and during the building of a formal ontology. Ontology on the Web requires more expressiveness. Classes are the focus of most ontology. Classes describe concepts in the domain. Slots describe properties of classes and instances Developing ontology includes [15]:

1. Defining classes in the ontology.
2. Arranging the classes in a taxonomic (subclass–super class) hierarchy.
3. Defining slots and describing allowed values for these slots.
4. Filling in the values for slots for instances.

A successful system. At present the construction of ontology is very much an art rather than a science [18]. The attempt is to formalize the ad-hoc process consists of the some basic steps [19].To supports methodology and to guide users step by step through the ontology engineering process an effective tool is desired. Along with the development of the methodology we therefore extended the core functionalities of Protégé.

III Ontology Building Tools

There is survey of Software Tools that have Ontology editing capabilities and are in use today. These ontology building tools ( Protégé 3.4, IsaViz, SWOOP and Apollo) may be useful for building ontology schemas (terminological component) alone or together with instance data. Concise descriptions of each software tool were compiled and then reviewed by the organization currently providing the software for commercial, open, or restricted distribution. The descriptions are factored into a dozen different categories covering important functions and features of the software. These categories are summarizing the results.
We have used only four “popular and accepted” ontology authoring tools (Apollo, Protégé 3.4, IsaViz and SWOOP), taking into consideration the advantages of these tools. Tools that provide support for the different phases of the ontology engineering process are referred to as ontology building tools. These tools are used for building a new ontology either from scratch or by reusing existing ontology, which usually supports editing, browsing, documentation, export and import from different formats, views; libraries and they may have attached inference engines, etc. [20]. The ontology editors are tools that allow users to visually manipulate, inspect, browse and code ontology and support in this way the ontology development and maintenance task [21]. In this section, we will provide a broad overview of some of the available ontology editor tools with a brief description of each tool, presenting the group that has developed it, its main features and functionalities, its URL etc.

IV PROTEGE

IV.1 Protege

Protege [6] is an ontology and knowledge base editor produced by Stanford University. Protégé is a tool that enables the construction of domain ontology, customized data entry forms to enter data. Protégé allows the definition of classes, class hierarchies, variables, variable-value restrictions, and the relationships between classes and the properties of these relationships. Protégé is free and can be downloaded from http://protege.stanford.edu [6]. Protege comes with visualization packages such as OntoViz, EZPal, etc.; all of these help the user visualize ontology with the help of diagrams. Stanford University is doing a magnificent job of continually improving Protégé. As part of its last update, Protégé now includes an interface for SWRL (Semantic Web Rule Language), which sits on top of OWL to do math, temporal reasoning, and adds Prolog-type reasoning rules. Stanford has a tutorial that covers the basics of using Protege with the OWL plug-in.

The strength of Protégé is that it supports at the same time tool builders, knowledge engineers and domain specialists. This is the main difference with existing tools, which are typically targeted at the knowledge engineer and lack flexibility for meta-modeling. This latter feature makes it easier to adapt Protégé to new requirements and/or changes in the model structure. When starting out on an ontology project, the first and reasonable reaction is to find a suitable ontology software editor [4]. These tools can help acquire, organize, and visualize the domain knowledge before and during the building of a formal ontology. The Protégé platform supports two main ways of modeling ontology: The Protégé-Frames editor enables users to build and populate ontology that are frame-based, in accordance with the Open Knowledgebase Connectivity Protocol (OKBC). In this model, an ontology consists of a set of classes organized in a subsumption hierarchy to represent a domain's salient concepts, a set of slots associated to classes to describe their properties and relationships, and a set of instances of those classes - individual examples of the concepts that hold specific values for their properties. The Protege-OWL editor enables users to build ontologies for the Semantic Web, in particular in the W3C's Web Ontology Language, "An OWL ontology may include descriptions of classes, properties and their instances. Given such ontology, the OWL formal semantics specifies how to derive its logical consequences, i.e. facts not literally present in the ontology, but entailed by the semantics. These entailments may be based on a single document or multiple distributed documents that have been combined using defined OWL mechanisms.

From the existing tools Protege 3.4 is chosen because it enables the construction of domain ontology, customized data entry forms to enter data. Protege allows the definition of classes, class hierarchies, variables, variable-value restrictions, and the relationships between classes and the properties of these relationships.

IV.1 IsaViz

IsaViz is a visual environment for browsing and authoring RDF models as graphs. This tool is offered by W3C Consortium. IsaViz [7] was developed by Emmanuel
Pietriga. The first version was developed in collaboration with Xerox Research Centre Europe which also contributed with XVTM, the ancestor of ZVTM (Zoom able Visual Transformation Machine) upon which IsaViz is built. As of October 2004, further developments are handled by INRIA Futures project In Situ.

Fig. 4 Screenshot of Protege

IsaViz also includes software developed by HP Labs (Jena 2 Semantic Web Toolkit), the Apache Software Foundation (Xerces Java 2), and makes use of the GraphViz library developed by AT&T Research. IsaViz does not follow or include any methodology for building ontology. IsaViz imports RDF/XML and N-Triples, and exports RDF/XML [13], N-Triples, Portable Network Graphics (PNG) and Scalable Vector Graphics (SVG). Therefore, it is possible to import ontology to other editors, for instance, Protégé or OilEd. The IsaViz environment is composed of four main windows: the IsaViz RDF,Editor window, the Graph window, the Definition window and the Attribute window.

4.3 Apollo

Apollo [8] is a user-friendly knowledge modeling application. The modeling is based around the basic primitives, such as classes, instances, functions, relations etc. Internal model is build as a frame system according to the internal model of the OKBC protocol.

Apollo’s class system is modeled according to the OKBC. The knowledge base consists of ontology’s that are hierarchically organized. Ontology can inherit other ontology’s and then use classes of inherited ontology’s as its own. Every ontology inherits at least one ontology – a default ontology, which contains all primitive classes: Boolean, integer, float, string, list etc. Class contains slots of two types: non template and template slots.

Apollo currently does not support non template class slots. For each class is possible to create a number of instances. An instance inherits all slots of the class. Each slot has a set of facets.

4.4 Swoop

SWOOP [9] is a Web-based OWL ontology editor and browser [4]. SWOOP contains OWL validation and offers various OWL presentation syntax views. It has reasoning support and provides a multiple ontology environment. Ontology can be compared, edited and merged. Different ontology can be compared against their Description Logic-based definitions, associated properties and instances. SWOOP’s interface has hyperlinked capabilities so that navigation can be simple and easy. SWOOP does not follow a methodology for ontology construction. Users can reuse external ontological data [4]. This is possible either by purely linking to the external entity, or importing the entire external ontology. It is not possible to do partial imports of OWL. There are several ways to achieve this, such as a brute-force syntactic scheme to copy/paste relevant parts (axioms) of the external ontology, or a more elegant solution that involves partitioning the external ontology while preserving its semantics and then reusing (importing) only the specific partition as desired. It is possible to search concepts across multiple ontologies. SWOOP makes use of an ontology search algorithm, that combines keywords with DL-based in order to find related concepts. This search is made along all the ontologies stored in the SWOOP knowledge base.

Fig. 3 Screenshot of Swoop
4.5 Ontoedit

OntoEdit is part of OntoStudio, based on IBM Eclipse framework. OntoEdit is a development environment for ontology design and maintenance. It supports multilingual development, and the knowledge model is related to frame-based languages. OntoEdit is based on an open plug-in structure. Every plug-in provides other features to deal with the requirements an ontology engineer has. Data about classes, properties, and individuals may be imported or exported via different formats, such as OXML, FLogic, RDF/RDFS, OWL and other formats supported by WebDav.

The professional version of the tool is available as a commercial product.

The Requirements Specification phase for ontology development leads to an ontology requirements specification document describing what an ontology should support. Collecting requirements for the envisaged ontology starts the ontology development. This task is performed by a team of experts for the domain accompanied by experts for modeling. It also guides an ontology engineer to decide about relevant concepts and their hierarchical structure in the ontology. This phase is supported by OntoEdit by the two plug-ins OntoKick and Mind2Onto5 for meta ontology description with automatically calculated statistic information.

![Fig. 3 Screenshot of Ontoedit](image)

**V EXPERIMENTAL RESULTS AND ANALYSIS**

The framework that we have set for analyzing all the tools are closely examined for the ontology development and then we compared all the tools against the evaluation framework. The result for comparison of tools are shown in the form of Tables which are classified on the basis of

<table>
<thead>
<tr>
<th>Feature</th>
<th>Protege 3.4</th>
<th>IsaViz</th>
<th>Apollo</th>
<th>SWOOP</th>
<th>Ontoedit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extensibility</td>
<td>Via plugins</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Via plugins</td>
</tr>
<tr>
<td>Ontology Storage</td>
<td>Files &amp; DBMS</td>
<td>Files</td>
<td>Files</td>
<td>Files</td>
<td>File</td>
</tr>
<tr>
<td>Availability</td>
<td>Free</td>
<td>Free</td>
<td>Free</td>
<td>Free</td>
<td>Free</td>
</tr>
</tbody>
</table>

**Table 1: Tools’ Architecture**

<table>
<thead>
<tr>
<th>Feature</th>
<th>Protege 3.4</th>
<th>IsaViz</th>
<th>Apollo</th>
<th>SWOOP</th>
<th>Ontoedit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Import Format</td>
<td>XML/RDF (S), XML Schema and OWL</td>
<td>XSLT, RDF(S), OIL, DAML+OIL, OWL</td>
<td>OCML, RDF(S), OIL, DAML</td>
<td>XML, RDF(S), FLogic and DAML+OIL</td>
<td></td>
</tr>
<tr>
<td>Export Format</td>
<td>XML, RDF (S), XML Schema, Java, html</td>
<td>XSLT, RDF(S), OIL, DAML+OIL, OWL</td>
<td>OCML, RDF(S), OIL, DAML</td>
<td>XML, RDF(S), FLogic and DAML+OIL</td>
<td></td>
</tr>
<tr>
<td>Merging</td>
<td>Via ANCHOR PROMPT plug-in</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>?</td>
</tr>
</tbody>
</table>

**Table 2: Tools’ interoperability**

<table>
<thead>
<tr>
<th>Feature</th>
<th>Protege 3.4</th>
<th>IsaViz</th>
<th>Apollo</th>
<th>SWOOP</th>
<th>Ontoedit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inference Engine</td>
<td>With PAL</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Exception Handling</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>NO</td>
</tr>
<tr>
<td>Consistency Checking</td>
<td>Via plug ins</td>
<td>Via Inheritance</td>
<td>Yes</td>
<td>Only checks</td>
<td>Yes</td>
</tr>
</tbody>
</table>

**Table 3: Tools’ inference services**

<table>
<thead>
<tr>
<th>Feature</th>
<th>Protege 3.4</th>
<th>IsaViz</th>
<th>Apollo</th>
<th>SWOOP</th>
<th>Ontoedit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Collaboration with other tools</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Ontology Library</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Visualization</td>
<td>No</td>
<td>Via plug-ins</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

**Table 4: Tools Usability**
5) Overview of Tools’ versioning and collaborative work support on the basis of Versioning and collaboration

<table>
<thead>
<tr>
<th>Feature</th>
<th>Protege 3.4</th>
<th>IsaViz</th>
<th>Apollo</th>
<th>Swoop</th>
<th>Ontoedit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Versioning</td>
<td>Supported</td>
<td>Not Supported</td>
<td>Not Supported</td>
<td>Not Supported</td>
<td>Not Supported</td>
</tr>
<tr>
<td>Collaboration</td>
<td>Not Fully Supported</td>
<td>Not Supported</td>
<td>Not Fully Supported</td>
<td>Not Supported</td>
<td>Not Fully Supported</td>
</tr>
</tbody>
</table>

Table 5: Tools Versioning & Collaboration

VI CONCLUSION AND FURTHER RESEARCH

For Ontology development, effective tools are a central requirement. Fortunately, software tools are already available to achieve most of the required activities of ontology development allowing us to focus specifically on the innovate requirements of ontology development within the model. Projects often involve solutions using numerous ontologies (Wine.rdf, food.owl, Companies.rdf etc.) from external sources. Sometimes there is also the need to use existing and newly developed in-house ontologies (i.e. camera.owl). For this reason it is important that the editing tools for ontology construction promote interoperability. Ontology Editors prove an asset in the development of ontologies. The need is to identify a suitable editor for a particular domain. A theoretical attempt has been made to analyze and make a comparative analysis of the various ontology editors available and their role in ontology building and maintenance. It can be further extended to choose and make use of an ontology editor for a particular domain ontology creation. To conclude, there are open source ontology tools (Protégé 3.4), there are ontology tools that demand learning/knowing a specific language (SWOOP) and there are ontology tools that are more graphic (IsaViz). Other tools are Web-based application (Apollo and SWOOP) or follow a methodology (Protégé 3.4 and SWOOP). Some tools only support common edition and browsing functionalities. Other tools provide ontology documentation, ontology import/export for different formats, graphical view of ontologies, ontology libraries and attached inference engines. It is quite clear Ontology development is an ad-hoc approach. Among several viable alternatives, one need to find which one would work better for the projected task that can easily and effectively be maintained and expressed. Though foundation of ontology is logic but it is a model of reality and the concepts in the ontology must reflect this reality. We have described a tool-assisted method for building the basis for ontologies adopted from domain analysis.

REFERENCES


AUTHOR’S PROFILE

Dr Suresh Jain
suresh.jain@rediffmail.com

C.S.Bhatia
is working as Assistant Professor in Sanghvi Innovative Academy, Indore. He has done ME(SW)from Institute of Engineering & Technology Devi Ahilya Vishwavidyalaya, Indore. He is Lifetime Member of ISTE. He Teaches Courses for B.Tech, M.Tech, M.E., BCA and MCA. His research areas of interest include NLP, AI, Mobile Computing, Human Computer Interaction, Semantic web Applications and ontologies. He has presented & published various research papers in national and international Journals and conferences.}

jeet.bhatia@rediffmail.com
Dharmendra Gupta
is working as Assistant Professor in Sanghvi Innovative Academy, Indore. He has done ME from SCS Devi Ahilya Vishwavidyalaya, Indore. He teaches courses for B.Tech, M.Tech, M.E., BCA and MCA. His research areas of interest include NLP, AI, Mobile Computing, Human Computer Interaction, Semantic web Applications and ontologies. He has presented & published various research papers in national and international Journals and conferences.
dhermendragupta86@gmail.com

Sumit Jain
is working as Assistant Professor in Sanghvi Innovative Academy, Indore. He has done ME from IET-DAVV, Indore. He teaches courses for B.Tech, M.Tech, M.E., BCA and MCA. His research areas of interest include DM, DW, NLP, AI, Mobile Computing, Human Computer Interaction, Semantic web Applications and ontologies. He has presented & published various research papers in national and international Journals and conferences.
suresh.jain@rediffmail.com

Bharat Pahadiya
is working as Assistant Professor in Sanghvi Innovative Academy, Indore. He has been pursuing ME from IES IPS Academy, Indore. He teaches courses for B.Tech, BCA. His research areas of interest include Networking, Software Engineering, NLP, AI, Mobile Computing, Human Computer Interaction, Semantic web Applications and ontologies. He has presented & published various research papers in national and international Journals and conferences.