2. Web Information Retrieval

“Text search engines are of course good for searching the text in documents, but the Semantic Web isn't text documents, it is data…”

“One thing to always remember is that the Web of the future will have BOTH documents and data. The Semantic Web will not supersede the current Web. They will coexist. The techniques for searching and surfing the different aspects will be different but will connect. Text search engines don't have to go out of fashion.”

- Tim Berners-Lee

This chapter introduces basic concepts, terminology, techniques and tools used throughout the thesis. It describes the evolution of the Web and different models that are used in Information Retrieval process. It also presents various tools and technologies applied in implementation.

2.1 Introduction

The Web is changing at a very fast pace, whether it be the content versatility or it be the technology that explores the web content in meaningful and useful information. Figure 2-1 shows the pyramid of the Web evolution (Spivack, 2007). The divisions on the pyramid represent the volume of information in each version of the Web. The right side shows the technologies that have been used, are being used or expected to be used in the future as shown as per the evolution time period shown on the left side of the pyramid.

The World Wide Web (Web 1.0) which is primarily based on hyperlinks requires keywords, co-occurrence and page rank for searching relevant web pages. The relevance of web pages in this face of the Web is usually computed using hubs and authorities (Kleinberg & Lawrence, 2001) or, keyword term frequency (Salton & Buckley, 1987). However, these techniques and other traditional search algorithms besides being simple and computationally sound, lack in searching semantically relevant web pages (Navigli & Velardi, 2003). This means that the pages that contain synonyms, hypernyms or hyponyms for the keywords rarely get incorporated during the search.
Further, Web 2.0, allows its users to interact and share their opinion through blogs, tagging, bookmarking sites, social networks etc. forming the social web altogether. Tagged web pages help in improving the search of relevant web pages on the Web, but inclusion of semantic knowledge (ontology and axiom) along with tagged web resources makes the search more fruitful.

In comparison to earlier Web versions, Semantic Web (Web 3.0) organizes data in the form of linked data instead of hyperlinked web pages. It uses multiple ontologies/RDFS/RDF/XML linked together to form a Giant Global Graph (GGG) (Berners-Lee T., 2007) through which the required data can be extracted semantically.

The intelligent web (Web 4.0) is believed to use reasoning based recommendations built on the ontologies and logical axioms (Farber, 2007). Although, Web 4.0 is termed as the next version of the semantic web (Spivack, 2007), the distinguished features of Web 4.0 have already been incorporated conceptually as a part of the Semantic Web Architecture (Berners-Lee, Hendler, & Lassila, 2001).
Retrieving and Organizing Web Resources Semantically for Informal E-Mentoring

(Hendler & Berners-Lee, 2010). Moreover, the future Web is believed to consist of the documents (all web pages linked through hyperlinks) as well as the data (the linked data in form of ontologies and RDFs) where every digital entity (documents and data) would be available to be explored collaboratively (socially). In addition, the techniques and methods for searching and retrieving relevant content would connect these versions (Berners-Lee T., 2009).

The process of retrieving useful information from the Web encompasses various Information Retrieval (IR) techniques varying with different Web versions. The work in the thesis deals with three versions of the Web; these are the WWW, the Social Web and the Semantic Web. The IR processes that are generally used for retrieving information from each type of the Web are discussed in the forthcoming sections.

The World Wide Web (WWW) being the first version of the Web is usually referred as ‘Web’ in early literature. However, in the thesis the Web has been used for the existing Web consisting of the WWW, the Social Web and the Semantic Web. To refer a specific version of the Web, these will be referred in particular. As mentioned earlier, the WWW is a vast collection of web pages interconnected with each other through web links called hyperlinks. Web page, a unit of information on the WWW has many synonyms like document, resource and page (Sebesta, 2007). These synonyms have been used interchangeably depending on the context in the thesis.

Information retrieval has attained new definitions with the advent of the Web. The web information retrieval deals with the representation, storage, organization of, and access to information items (Baeza-Yates & Ribeiro-Neto, 1999). Low cost, greater access, publishing freedom and linking documents to many other documents on the Web are the primary reasons for the popularity of the Web as a highly interactive medium and immeasurable source of information. Searching useful information to users’ interest in the ever-growing volume of the Web is a real challenge for Web information retrieval research. Here we discuss a few basic concepts and the models in this field related to the thesis work.
2.2 The World Wide Web

World Wide Web (WWW) is a collection of documents which are interlinked through the hypertext. These interlinked hypertext web documents are accessed via the Internet with the help of a web browser. These documents or web pages may contain text, images, videos and other multimedia, and hyperlinks through which the navigation among documents is possible. HTTP (Hypertext Transfer Protocol), a TCP/IP based communication protocol is used to deliver web resources virtually on the WWW.

2.2.1 Information Retrieval from the WWW

In a conventional information retrieval system a document is described logically as a collection of index terms. An index term is a keyword which has some meaning of its own such as nouns. In general, the index term may consist of all words in text of the document. However, considering index in this way raises concerns over the text semantics. This issue has been discussed many times in the information retrieval literature. These index terms are compared to find similarity or relevance of the document to a query using various models.

The web retrieval process can be explored in one of the two operational modes, ad-hoc and filtering. In ad-hoc retrieval, the documents in the collection remain relatively static while new queries are submitted to the system. In the other mode, the queries relatively remain static while new documents come in the system (and/or leave the system). This operational mode is termed as filtering. The work in the thesis belongs to the filtering task.

In filtering, the ranking of documents is based on the users’ information need, which is usually constructed through a set of keywords provided either by a user explicitly or extracted implicitly through some preferred relevant documents. This initial information need to improve searching is sometimes referred as ‘user profile’ or expansion of the query/topic which takes care of a user’s information needs. The simplistic way to construct the expanded topic list is to ask user to provide keywords related to his/her search requirement. In some cases the user is also asked to provide relevance feedback about the searched documents to build a training set (consisting of
two sets of relevant and non-relevant documents) to be used for improving future retrieval results. Though this approach is simple, it requires a user to provide lot of details that describes his/her profile. Moreover, the user is expected to be familiar with the search topic. (S)he has to provide related keywords or required to be able to judge the relevance of documents. The work in the thesis has adopted an approach to construct an expanded topic list for filtering by using semantic knowledge on a search topic. Constructing expanded topic list using semantic structure (consisting of the search topic and its useful related concepts) has a number of benefits towards the filtering task as compared to the above mentioned method. The semantic structure based topic expansion methods alleviates the need for the user to provide related keywords on the required search topic and neither the user is required to spend time in constructing the training set. Context sensitive document retrieval is the added benefit of the semantic structure based filtering.

**Classic Model:**

In the retrieval process, documents are retrieved based on the relevance prediction i.e. *similarity of documents* to a search query. This similarity can be determined using various models listed in Figure 2-2. Boolean, Probabilistic and
Vector models are three classic models used in Information Retrieval for predicting similarity between documents and the search query. It is significant here to notice that in the IR models (Boolean, Probabilistic, vector etc.), the logical view of documents and the basis for constructing the search topic expansion (lexical or semantic) are orthogonal aspects of the retrieval system. This means that defining a document and expanding a search topic are mutually exclusive subprocesses and therefore different models can be applied on them. A brief introduction, advantages and disadvantages of each classic model is given below.

In **Boolean model** documents and queries are represented as sets of index terms and therefore are referred as Set Theoretic (Grossman & Frieder, 2004), which extends to fuzzy and ‘Extended Boolean’. For the Boolean model (Lashkari, Fereshteh, & Vahid, 2009), the index term weight variables are all binary i.e., if the document index term exists in the query then the weight is 1 else it is 0. The simple and clear formalism is the main advantage. However, the document relevance being measured in binary terms, it produces exact matching. As a result the Boolean model retrieves either a small number of documents or a large number of documents.

The framework for modeling document and query representations based on probability theory is named as **Probabilistic Model**. The probabilistic model attempts to estimate the probability that the user will find the document \( d_j \) relevant. This model assumes that this probability of relevance depends on a given user query \( q \) and the document representations only. Based on this assumption it separates out the relevant and non relevant documents into two sets. The model also assumes a pre-existing subset of all documents which the user prefers as the answer set for the query. The disadvantages and the troublesome issues in this model include the need to guess the initial separation of documents into relevant and non relevant, and that the method does not take into account the frequency with which an index term occurs inside a document. The only advantage is that it produces a list of documents ranked in decreasing order of their probability of being relevant.

In the **VSM**, documents and queries are represented as vectors in an n-dimensional space where the terms in the document or query represents each dimension. Therefore the model is termed as ‘Algebraic’. It overcomes the
disadvantages of the Boolean and Probabilistic Models by assigning non-binary weights to the index terms in queries and in documents. These term weights are used to compute the degree of similarity between document and the user query. Unlike Boolean model, the Vector Space Model (VSM) computes the document relevance which varies from 0 to +1, rather than only either 0 or 1. Thus, instead attempting to predict a document is relevant or not, the vector model ranks the documents according to their degree of similarity to the query. Moreover, The vector model proposes to evaluate the degree of similarity of document $d_j$ with regard to the query $q$ as the correlation between the vectors $\vec{d}_j$ and $\vec{q}$. This correlation can be quantified by the cosine of angles between two vectors. A document can be retrieved even if it matches the query partially.

The main advantages of the VSM include: i) improved retrieval performance due to its term-weighting scheme, ii) the retrieval of documents based on partial matching and iii) sorted order of the documents according to the degree of similarity to the query, using cosine ranking formula. The most important part in this model is the way the index terms are assigned the weight. One of the widely used schemes for assigning weights to the index terms is called $tf-idf$ (term frequency – inverse document frequency). The ranked answer sets can be improved by introducing query expansion or relevance feedback. In literature a large variety of alternative ranking strategies have been compared to the vector model, but the consensus seems to be that, the vector model is either superior or equivalently good to the known alternatives. Moreover, it is fast and simple. Therefore, the vector model is a popular and widely used by the research community for information retrieval.

**Structured Models:**

The structured text retrieval models deal with the retrieval of documents based not only the text of the document but also the properties associated with the text such as the index term with bold or italic font, term in heading or in subsection etc. Such retrieval systems search all documents that satisfy the query conditions. **Non-Overlapping Lists** and **Proximal Nodes** are two methods used by the model for structure based documents retrieval.
In **Non-Overlapping Lists**, the whole text in each document can be divided in non-overlapping text regions which are collected in a list. The collection of these lists may correspond to various features of the documents, for instance, list of all chapters in the document, a second list of all sections in the document, and a third list of all subsections in the document. These lists are kept as separate and distinct data structures, for keyword matching.

**Proximal Nodes** allows the definition of independent hierarchical indexing structures over the same document text. Each of these indexing structures is a hierarchy composed of chapters, sections, subsections, paragraphs, pages and lines which are called nodes. These nodes are associated to different text regions. However, two distinct hierarchies might refer to overlapping text regions. For a given query term, these lists or hierarchies are searched which results in a faster query processing.

**Semantic Structure:**

In an Information Retrieval system document vectors are compared with query vectors to predict the relevance ranking. For this purpose, index terms in documents and queries are matched and weighted. The semantic structure allows a machine to understand meaning of a search query and, to contextually associate the content of a document to a search query. The use of semantics based information retrieval alleviates the need of generating users’ explicit requirements for the search and the relevance feedback of the documents as it was in the case of probability model. Therefore, the proposed system in this thesis makes use of the semantic structure along with the VSM.

### 2.3 The Social Web

The Social Web is a set of relationships that link together people over the Web (Appelquist, et al., 2010). Among many social networking sites like Facebook\(^4\), MySpace\(^5\), YouTube\(^6\) etc., there exist Social Bookmarking Sites (SBS) also, such as

---

5 [in.myspace.com/](http://in.myspace.com/)
6 [www.youtube.com/](http://www.youtube.com/)
digg\textsuperscript{7}, StumbleUpon\textsuperscript{8}, reddit\textsuperscript{9}, delicious\textsuperscript{10}, DZone\textsuperscript{11} and so on, to name a few. A SBS is a social site where web users are not only allowed to tag their web resources and organize them but they are also allowed to share others’ bookmarks. While searching web resources on the Web, web users bookmark the web pages of their interest and tag them with the keywords that describes about the content of the web page. Tagging in this way helps web users by directly reaching to their previously chosen and stored links at any time, without spending time on repetitive web search.

### 2.3.1 Information Retrieval from the Social Web

The information in SBS (Chi & Mytkowicz, 2008) is structurally perceived as a set of bookmarks, $B$, where each bookmark, $b \in B$, is a set of triples: ${r, u, t_1}$, ${r, u, t_2}$, …, ${r, u, t_k}$. Thus a single triple consists of three data units, $< r, u, t_k >$, signifying a web resource $r_i$ being tagged by user $u_j$ with tag $t_k$. Each bookmark $b$ represents a single web resource. The web resource is tagged by several users using various keywords as tags relevant to the resource content. It can further be noticed that a web resource $r$ which actually represents a URL on the WWW may or may not be tagged in a SBS. Thus only those web resources could be retrieved from a portal (site) which has been tagged by at least one web user.

Extraction of information from a portal which is also called as deep web search needs the knowledge of the portal’s web pages structure and/or the key to its search mechanism. In such type of information retrieval, the crawlers need to be designed according to the portal’s structure.

The work in this thesis uses a popular SBS, delicious.com to extract tagged web resources by using the proposed social semantic focused crawler and in predicting social semantic relevance of these retrieved documents. The above notations will be used to describe the retrieval and ranking approaches which relate to the tagged web resources. These approaches are explained in chapter 4 and chapter 5.

---

\textsuperscript{7} www.digg.com/
\textsuperscript{8} www.stumbleupon.com/
\textsuperscript{9} www.reddit.com/
\textsuperscript{10} www.http://delicious.com/
\textsuperscript{11} www.dzone.com/
2.4 The Semantic Web

The Semantic Web is a web of linked data whereas the WWW which is a web of hyperlinked documents. E-Learning characteristics can effectively be achieved through the new Web architecture, Semantic Web (SW) (Berners-Lee, Hendler, & Lassila, 2001). Research (Drucker, 2000); (Alsultanny, 2000); (Stojanovic, Staab, & Studer, 2001); (Rosic, Stankov, & Glavinic, 2002); (Hatem, Ramadan, & Neagu, 2005); (Sylvain & Catherine, 2006) shows the benefits of using semantic web as a platform for e-Learning and therefore for gathering information, semantic content organization and retrieval for E-mentoring.

2.4.1 Semantic Web Architecture

Conceptually the SW represents a layered architecture (Berners-Lee T., 2006) (Figure 2-3), where each level provides different degree of expressiveness. The bottom most layer of the architecture allows to provide a definition for controlled vocabulary and namespaces. Next layer, Resource Description Framework (RDF) represents information about resources in the WWW or the linked data on the Web. It consists of a triplet – resource, property and value. Ontology vocabulary, the next upper layer, adds more semantics to RDF by allowing a better specification of constraints on concepts. This middle layer and the backbone of Semantic web, forms knowledge about a certain domain by providing relevant concepts and relations.

Figure 2-3: Semantic Web Layered Architecture proposed by Tim Berners-Lee
between them. Often cited definition of ontology defines it as “an explicit specification of a conceptualization” (Gruber T. R., 1993) i.e., an ontology defines (specifies) the concepts, relationships, and other distinctions that are relevant for modeling a domain. The specification takes a form of representational vocabulary (classes, relations, and so forth) which provides meaning for the vocabulary and formal constraints on its coherent use (Gruber T., 2008). RIF is an acronym for Rule Interchange Format, a specification for a rule language using RDF Schema as its ontology basis (Gerber, Merwe, & Barnard, 2008). The rules for explicit inference are established by Logic layer which are then executed by the Proof layer. The top layers, Trust provide mechanisms to imbibe trust in a given proof (Bittencourt, Costa, Soares, & Pedro, 2008).

**2.4.2 Information Retrieval from the Semantic Web**

Information from the Semantic Web can be retrieved using the Linked Data. The Linked Data can be visualized as a published knowledge base and a part of the Semantic Web enabled applications. The individually created knowledge base by different organizations or groups can be published on the Linked Open Data cloud\(^\text{12}\) (Figure 2-4) which resides on the Web. The ontology developed as part of the presented work can also be a part of this web.

**Linked Data** refers to data published on the Web in such a way that it is machine-readable, its meaning is explicitly defined, it is linked to other external data sets, and can in turn be linked to from external data sets (Bizer, Heath, & Berners-Lee, 2009). Berners-Lee (2006) outlined 'Linked Data principles' for publishing the data on the Web such that all published data becomes part of a single Giant Global Graph (GGG) (Berners-Lee T., 2007). These principles are the rules for publishing and connecting data using the infrastructure of the Web while adhering to its architecture and standards. The principles are as follows:

1. Use URIs as names for things
2. Use HTTP URIs so that people can look up those names
3. When someone looks up a URI, provide useful information, using the standards

4. Include links to other URIs, so that they can discover more things.

The data on the Semantic Web is described and linked using Resource Description Framework (RDF) and/or Web Ontology Language (OWL).

**RDF**

RDF provides a generic, abstract data model for describing resources using subject, predicate, object triples. A set of such triples is called an RDF graph (Klyne & Carroll, 2004). The triple in the Figure 2-5 can be represented as <Anjali, hasColleague, Richa>. Everything in RDF is referenced by a unique URI (Universal Resource Identifier). RDF has XML syntax where its tags describe a specific meaning. As an extension to the example illustrated in Figure 2-5, the RDF code can be represented as URI based triples as following:
Here every ‘description’ element describes a ‘resource’ and every attribute or nested element inside a ‘description’ is a property of that ‘resource’ (Bechhofer, Horrocks, & Patel-Schneider, 2003). The resources are referred by using URIs\(^\text{13}\).

RDF defines relationship between two entities via predicates. However, it does not provide any domain-specific terms for describing classes of things in the world and how they relate to each other. This function is served by taxonomies, vocabularies and ontologies expressed in RDFS (the RDF Vocabulary Description Language, also known as RDF Schema) and OWL (Heath & Bizer, 2011).

**RDFS**

RDFS is a language for describing lightweight ontologies in RDF. These are often referred to as vocabularies. RDFS primarily extends RDF by schema vocabulary. In its most simple form, RDFS vocabularies consist of class and property type definitions, such as the class ‘Concept’ (of which there may be many

\(^{13}\)http://www.w3.org/Addressing/
instances) and the property ‘hasSubconcept’. The primitives of the RDFS language are defined in two separate namespaces:

- The <http://www.w3.org/2000/01/rdf-schema#> namespace is associated (by convention) with the rdfs: namespace prefix
- The <http://www.w3.org/1999/02/22-rdf-syntax-ns#> namespace is associated (by convention) with the ‘rdf:’ namespace prefix

The two basic classes within the RDFS language are:

- rdfs:Class which is the class of resources that are RDF classes,
- rdf:Property which is the class of all RDF properties

All schema information in RDFS is defined with RDF triples. It provides formal description of inference rules that work on the pattern of triples. It makes use of vocabularies such as class, property, type, subClassOf, domain and range to define the relationships among data in RDF.

**Ontology**

Ontology is a fundamental and important technology for Semantic Web (Kozaki, Hayashi, Sasajima, Tarumi, & Mizoguchi, 2008). It has the potential to construct a dynamic and semantic structure of e-Content that can be shared, reused and enhanced with trust for learning purposes (Mizoguchi & Bourdeau, 2000).

Ontology defines (specifies) the concepts, relationships, and other distinctions that are self explanatory for modeling a domain (Gruber T. R., 1993); (Gruber T. R., 1995). For the purpose of defining semantic relations among various terms or concepts, ontology provides an efficient solution. Applications like e-Learning involve various abstract relations that are defined among different concepts for building the knowledge base. Instead of using hypernyms, hyponyms, synonyms or antonyms from WordNet\(^{14}\) which is a taxonomic hierarchy of natural language terms, the structure consists of conceptually related terms such as sub concepts, super concepts and sibling concepts, which represent a more significant and usable description of a concept.

\(^{14}\) http://wordnet.princeton.edu/
Mathematically, a core Ontology is defined as a structure, (Bozsak, et al., 2002),

\[ O := (C, \leq_C, R, \sigma_R, \leq_R, A, \sigma_A, T) \]

which consists of:

i) four disjoint sets \( C, R, A \) and \( T \) whose elements are called concept identifiers, relation identifiers, attribute identifiers and data types, respectively.

ii) a partial order \( \leq_C \) on, \( C \) called concept hierarchy or taxonomy,

iii) a function \( \sigma_R: R \rightarrow C^+ \) called relation signature,

iv) a partial order \( \leq_R \) on, \( R \), called relation hierarchy, where \( r_1 \leq_R r_2 \) implies \( |\sigma_R(r_1)| = |\sigma_R(r_2)| \) and \( \pi_i(\sigma_R(r_1)) = \pi_i(\sigma_R(r_2)) \) for each \( 1 \leq i \leq |\sigma_R(r_1)| \), \( (\pi_i(t)) \) is the \( i \)th component of tuple \( t \),

v) a function \( \sigma_A: A \rightarrow C \times T \), called attribute signature, and

vi) a set \( T \) of datatypes such as strings, integers, etc.

This kind of a graph based hierarchical structure defines a clear association of concepts that helps in simplifying the reasoning tasks among them which makes the ontology suitable for semantically organizing e-Learning resources. For e-Learning purpose ontologies are desired to:

- define various concepts under different educational domains, and
- organize various web repositories that are relevant to the concepts from educational domain.

This provides a separate structure for concepts (topics) and content (learning repositories), and consequently simplifies the process of ontology maintenance and incorporation of frequent changes. Knowledge base adds more semantic data to the ontologies.

**Knowledge Base** is defined as a structure:

\[ KB := (C_{KB}, R_{KB}, I, \iota_C, \iota_R) \]

consisting of:
i) two sets $C_{KB}$ and $R_{KB}$.

ii) a set $I$ whose elements are called instance identifiers (or instances or objects for short),

iii) a function $t_c: C_{KB} \rightarrow \beta(I)$ called concept instantiation,

iv) a function $t_r: R_{KB} \rightarrow \beta(I^+)$ called relation instantiation.

The representation of Ontology for Web which can be interpreted and accessed by humans and machines is written in the Web Ontology Language (OWL). It is explained in more detail in section 2.5. The next sub-section deals with the ontology construction and introduction of the tools to access it.

### 2.4.3 Building and Accessing Ontology

Though there are a number of tools and techniques to retrieve different types of content from the Web, we would discuss here only those which have been used in the thesis. At the same time, the reasons of choosing the tools and/or techniques, and the alternate available options will be discussed too.

Construction of Ontology initially requires several iterations before finalizing the representation of educational concepts in a graphical (linked) form. Real world concepts’ conceptualization is comparatively simpler, due to the intuitive perception of real world relations. For concepts belonging to academia, the conceptualization requires in-depth knowledge and experts’ consensus over a domain.

In general, the first step in building ontology requires putting the concepts and the relations among them on the paper. After some iterations and reaching to an acceptable structure, the construction of schema can be outlined. This schema thereafter will provide the rules and meanings to the concepts in ontology. Additionally the axioms are specified on this schema to enable reasoning and preventing the formation of inconsistent relationships among concepts. The process of constructing ontology till this state can be considered as the design of ontology which needs manual intervention. This ontology design must be transformed to an interpretable and understandable web language for humans and machines. This task can be achieved through a standard Web Ontology Language called OWL.
Evolving from earlier proposals (Hogan, 2011) for Web ontology languages such as that of SHOE (Heflin, Hendler, & Luke, 1999), DAML (Hendler & McGuinness, 2000), OIL (Fensel, Horrocks, McGuinness, & Patel-Schneider, 2001) and the subsequent hybrid DAML+OIL (McGuinness, Fikes, Hendler, & Stein, 2002) in 2001 the W3C began working on a new ontological language that was built upon RDFS with more expressive semantics, enabling richer entailment regimes. In 2004, the resulting Web Ontology Language (OWL) was recognized as a W3C Recommendation (McGuinness & Harmelen, 2004).

Ontology in OWL can be manually constructed using an ontology editor such as ‘protégé’. There are several open source tools available that enable ontology accessing through APIs. One of such tool is protégé OWL API and Jena engine. These tools and APIs are described below.

**OWL: Web Ontology Language**

The OWL (Web Ontology Language), a W3C recommendation is designed to be used by applications/machine that requires processing of the content instead of just presenting information to human users. OWL facilitates greater machine interpretability of web content than that supported by XML, RDF, and RDF Schema (RDF-S). It provides additional vocabulary along with a formal semantics (McGuinness & Harmelen, 2004). OWL has three increasingly-expressive sublanguages: OWL Lite, OWL DL, and OWL Full. OWL Lite supports primarily those users who need a classification hierarchy and simple constraints. For example, while it supports cardinality constraints, it only permits cardinality values of 0 or 1. OWL DL includes all OWL language constructs, but they can be used only under certain restrictions (for example, while a class may be a subclass of many classes, a class cannot be an instance of another class). OWL DL is named ‘DL’ due to its correspondence with description logics, a field of research that has studied the logics and forms the formal foundation of OWL. In OWL Full a class can be treated simultaneously as a collection of individuals and as an individual in its own right. OWL Full allows an ontology to augment the meaning of the pre-defined (RDF or OWL) vocabulary. However, it is unlikely that any reasoning software will be able to support complete reasoning for every feature of OWL Full.
An OWL document consists of optional ontology headers plus any number of class axioms, property axioms, and facts about individuals. Some basic OWL constructs (Bechhofer, et al., 2004) are summarized in Table 2-1.

### Table 2-1: OWL constructs summarization

<table>
<thead>
<tr>
<th>OWL Constructs</th>
<th>Description</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Class descriptions</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OWL Class name</td>
<td>a class identifier (a URI reference)</td>
<td><code>&lt;owl:Class rdf:ID=&quot;Faculty&quot;/&gt;</code></td>
</tr>
<tr>
<td>enumeration</td>
<td>contains exactly the enumerated individuals, defined with the <code>owl:oneOf</code> property</td>
<td><code>&lt;owl:Class&gt; &lt;owl:oneOf rdf:parseType=&quot;PostStatus&quot;&gt; &lt;owl:Thing rdf:about=&quot;#Permanent&quot;/&gt; &lt;owl:Thing rdf:about=&quot;#Adhoc&quot;/&gt; &lt;/owl:oneOf&gt; &lt;/owl:Class&gt;</code></td>
</tr>
<tr>
<td>property restriction</td>
<td>a class of all individuals which satisfy a particular property restriction</td>
<td><code>&lt;Constructs on property are defined separately in the Table&gt;</code></td>
</tr>
<tr>
<td>intersection</td>
<td>a class that satisfies boolean AND of class descriptions</td>
<td><code>&lt;owl:intersectionOf&gt;</code> Results in a set which is common to Classes</td>
</tr>
<tr>
<td>union</td>
<td>a class that satisfies boolean OR of class descriptions</td>
<td><code>&lt;owl:unionOf&gt;</code> statement describes an anonymous class for which the class extension contains those individuals that occur in at least one of the class extensions of the class descriptions in the list</td>
</tr>
<tr>
<td>complement</td>
<td>a class that satisfies boolean NOT of class descriptions</td>
<td><code>&lt;owl:complementOf&gt;</code> statement describes a class for which the class extension contains exactly those individuals that do not belong to the class extension of the class description that is the object of the statement.</td>
</tr>
<tr>
<td><strong>Simple Properties</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DatatypeProperty</td>
<td>Binary relations between instances of classes and RDF literals and XML Schema datatypes</td>
<td><code>&lt;owl:ObjectProperty rdf:ID=&quot;locatedIn&quot;&gt; ... &lt;rdfs:domain rdf:resource=&quot;http://...owl#Thing&quot;/&gt; &lt;rdfs:range rdf:resource=&quot;#Region&quot;/&gt; &lt;/owl:ObjectProperty&gt;</code></td>
</tr>
<tr>
<td>ObjectProperty</td>
<td>Binary relations between instances of two classes</td>
<td><code>&lt;owl:ObjectProperty rdf:ID=&quot;teachesIn&quot;&gt; &lt;rdfs:domain rdf:resource=&quot;#Faculty&quot;/&gt; &lt;rdfs:range rdf:resource=&quot;#College&quot;/&gt; &lt;/owl:ObjectProperty&gt;</code></td>
</tr>
<tr>
<td>rdfs:domain</td>
<td>Used to restrict the ‘from’ relation</td>
<td><code>&lt;Faculty, teachesIn, College&gt;</code> Restricting domain as Faculty, for ‘teachesIn’ property</td>
</tr>
</tbody>
</table>
## OWL Constructs

<table>
<thead>
<tr>
<th>OWL Constructs</th>
<th>Description</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>rdfs:range</td>
<td>Used to restrict the ‘to’ relation</td>
<td>&lt;Faculty, teachesIn, College&gt;</td>
</tr>
<tr>
<td></td>
<td>The individuals of the class College can be included in the range (as object) of the property ‘teachesIn’.</td>
<td></td>
</tr>
<tr>
<td>rdfs:subPropertyOf</td>
<td>The property can be defined to be a specialization (subproperty) of an existing property.</td>
<td>&lt;owl:ObjectProperty rdf:ID=&quot;teachesIn&quot;&gt; &lt;rdfs:subPropertyOf rdf:resource=&quot;#associatedWith&quot; /&gt; &lt;rdfs:range rdf:resource=&quot;#WineColor&quot; /&gt; ... <a href="">owl:ObjectProperty</a></td>
</tr>
</tbody>
</table>

## Property Characteristics

<table>
<thead>
<tr>
<th>Property Characteristics</th>
<th>Description</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>TransitiveProperty</td>
<td>If a property, P, is specified as transitive then for any x, y, and z: P(x,y) and P(y,z) implies P(x,z)</td>
<td>&lt;rdf:type rdf:resource=&quot;&amp;owl;TransitiveProperty&quot; /&gt;</td>
</tr>
<tr>
<td>SymmetricProperty</td>
<td>If a property, P, is tagged as symmetric then for any x and y: P(x,y) iff P(y,x)</td>
<td>&lt;rdf:type rdf:resource=&quot;&amp;owl;SymmetricProperty&quot; /&gt;</td>
</tr>
<tr>
<td>FunctionalProperty</td>
<td>If a property, P, is tagged as functional then for all x, y, and z: P(x,y) and P(x,z) implies y = z</td>
<td>&lt;rdf:type rdf:resource=&quot;&amp;owl;FunctionalProperty&quot; /&gt;</td>
</tr>
<tr>
<td>inverseOf</td>
<td>If a property, P1, is tagged as the owl:inverseOf P2, then for all x and y: P1(x,y) iff P2(y,x)</td>
<td>&lt;rdf:type rdf:resource=&quot;&amp;owl;FunctionalProperty&quot; /&gt;</td>
</tr>
<tr>
<td>InverseFunctionalProperty</td>
<td>If a property, P, is tagged as InverseFunctional then for all x, y and z: P(y,x) and P(z,x) implies y = z</td>
<td>&lt;rdf:type rdf:resource=&quot;&amp;owl;InverseFunctionalProperty&quot; /&gt;</td>
</tr>
</tbody>
</table>

## Property Restrictions

<table>
<thead>
<tr>
<th>Property Restrictions</th>
<th>Description</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>owl:allValuesFrom</td>
<td>The individuals for which all values of the</td>
<td><a href="">owl:Restriction</a> &lt;owl:onProperty rdf:resource=&quot;#teaches&quot; /&gt; &lt;owl:allValuesFrom rdf:resource=&quot;#Faculty&quot; /&gt;</td>
</tr>
<tr>
<td>OWL Constructs</td>
<td>Description</td>
<td>Examples</td>
</tr>
<tr>
<td>----------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>-----------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>property P come from class C.</td>
<td></td>
<td><a href="">owl:Restriction</a></td>
</tr>
</tbody>
</table>
| owl:someValuesFrom   | All individuals for which at least one value of the property P comes from class C | <owl:Restriction>  
<owl:onProperty rdf:resource="# associatedWith" />  
<owl:allValuesFrom rdf:resource="# Faculty" />  
</owl:Restriction>|
| owl:hasValue         | All individuals that have the value A for the property P.                    | <owl:Restriction>  
<owl:onProperty rdf:resource="#hasSugar" />  
<owl:hasValue rdf:resource="#Dry" />  
</owl:Restriction>|
| owl:maxCardinality   | It can be used to specify an upper bound.                                   | <owl:Restriction>  
<owl:onProperty rdf:resource="#hasParent" />  
<owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">2</owl:cardinality>  
</owl:Restriction>|
| owl:minCardinality   | It can be used to specify a lower bound. In combination, the two can be used to limit the property's cardinality to a numeric interval. | <owl:Restriction>  
<owl:onProperty rdf:resource="#hasParent" />  
<owl:minCardinality rdf:datatype="&xsd;nonNegativeInteger">2</owl:cardinality>  
</owl:Restriction>|
| Class Axioms         |                                                                               |                                                                                               |
| rdfs:subClassOf      | The class extension of a class description is a subset of the class extension of another class description. | <owl:Class rdf:ID="College">  
<rdfs:subClassOf rdf:resource="#University" />  
</owl:Class>|
| owl:equivalentClass  | A class description has exactly the same class extension as another class description. | <owl:Class rdf:about="#US_President">  
<owl:equivalentClass rdf:resource="#PrplResOfWhiteHouse"/>  
</owl:Class>|
| owl:disjointWith     | The class extension of a class description has no members in common with the class extension of another class description. | <owl:Class rdf:about="#Man">  
<owl:disjointWith rdf:resource="#Woman"/>  
</owl:Class>
Protégé OWL editor

Protégé\footnote{http://protege.stanford.edu/} is an open source ontology editing tool. Protégé platform supports two main ways of modeling ontologies. The Protégé-Frames editor enables users to build and populate ontologies that are frame-based, and the Protégé-OWL editor enables users to build ontologies for the Semantic Web, particularly in OWL. At its core, Protégé implements a rich set of knowledge-modeling structures and actions that support the creation, visualization, and manipulation of ontologies in various representation formats. It provides easy to use interface to build OWL compatible ontology and put assertions on the classes and properties defined in the ontology. Further, Protégé can be extended via a plug-in architecture and a Java-based Application Programming Interface (API), specifically for building knowledge-based tools and applications. The work in the thesis has been carried out on the Protégé OWL editor, version 4.1.

The Protégé-OWL editor enables users to load and save OWL and RDF ontologies; edit and visualize classes and properties; define logical class characteristics as OWL expressions; execute reasoners such as description logic classifiers; and edits OWL individuals for Semantic Web markup.

According to a survey (Denny, 2004) descriptions of 94 ontology editors are available to the ontology building community. The results of which are categorized and available on the site\footnote{http://www.xml.com/2004/07/14/examples/Ontology_Editor_Survey_2004_Table_-_Michael_Denny.pdf}. However, the Protégé OWL editor is one the most popular editor among researchers due to the active online community support, easy to use interface for handling OWL ontology and its availability as open source. Moreover, it provides Protégé OWL API to enable ontology access through a programming language.

Protégé OWL API

The Protégé-OWL API\footnote{http://protege.stanford.edu/plugins/owl/api/} is an open-source Java library for the Web Ontology Language and RDF(S). The API provides classes and methods to load and save OWL files, to query and manipulate OWL data models, and to perform reasoning.
OWL API implements Jena at its core structure.

**Jena**

Jena\(^{18}\) is a Java framework for building Semantic Web applications. It provides a programmatic environment for RDF, RDFS and OWL, SPARQL and includes a rule-based inference engine. In the presented work Jena has been used to execute SPARQL queries and in a few operations to update the ontology.

**2.4.4 Accessing Linked Data**

The data from the Linked Data graph can be retrieved by one of the three means given below (Heath & Bizer, 2011).

1. LDspider\(^{19}\) (Isele, Harth, Umbrich, & Bizer, 2010), a Linked Data crawler that can process a variety of Web data formats including RDF/XML, Turtle, Notation 3, RDFa and many microformats. LDspider supports different crawling strategies and allows crawled data to be stored either in files or in an RDF store (via SPARQL/Update).
2. Jena TDB\(^{20}\), an RDF store which allows data to be added using SPARQL/Update and querying the data afterwards using the SPARQL query language.
3. SPARQL End points with Jena for Semantic Web.

The work in the thesis has used the third option to retrieve data from Dbpedia i.e., using Jena and SPARQL end points and a brief introduction of these tools is given below.

**SPARQL**

SPARQL\(^{21}\), an acronym for SPARQL Protocol And RDF Query Language is a W3C standard and is currently at version 1.1. It is a query language (Harris & Seaborne, 2012) for RDF which is a directed, labeled graph data format for representing information in the Web of data. SPARQL can be used to express queries across diverse data sources, whether the data is stored natively as RDF or viewed as


\(^{19}\) [http://code.google.com/p/ldspider/](http://code.google.com/p/ldspider/)

\(^{20}\) [http://openjena.org/TDB/](http://openjena.org/TDB/)

\(^{21}\) [http://www.w3.org/TR/sparql11-query/](http://www.w3.org/TR/sparql11-query/)
RDF via middleware. The results of SPARQL queries can be result sets or RDF graphs. It supports four types of queries: SELECT query (to retrieve a sub graph from the RDF satisfying the filter conditions); ASK query (to ask whether there exist matches to the query or not); CONSTRUCT query (to construct a RDF triple or graph); and DESCRIBE query (to describe the resource matched by the given variable.

**An Example of a SPARQL query**

```sparql
String query="PREFIX dbp: <http://dbpedia.org/resource/>" +
   "PREFIX dbp2: <http://dbpedia.org/ontology/>" +
   "SELECT ?abstract " +
   "WHERE {" +
   "dbp:+ q +" dbp2:abstract ?abstract . " +
   "FILTER langMatches(lang(?abstract), 'en')" +
   "}";
```

The example creates a query string that extracts an abstract from dbpedia ontology (linked data) for a given resource (here concept).

**SPARQL End Points**

SPARQL endpoints are web services provided remotely by the sources of Linked Data for their dataset through SPARQL Protocol. The results can be obtained in various formats including XML, JSON, RDF, N3, NTriples or plain text. If the datasets have their endpoint to the published data on the Linked Data, then the results from multiple datasets can also be accessed. For example, to access data using the dbpedia endpoint, an appropriate SPARQL query can be created using Jena as following:

```java
com.hp.hpl.jena.query.Query q = QueryFactory.create(query);
```

In this example statement ‘q’ is the Jena query object, where as ‘query’ is a SPARQL query string as it has been presented in the example.

```java
QueryExecution qexec = QueryExecutionFactory.sparqlService("http://dbpedia.org/sparql", q);
```

The above statement initializes the query execution with the remote service using the API sparqlService.
2.5 Multi Agent Systems

A software agent is a program situated in some environment such that the agent is capable of autonomous action in this environment in order to meet its design objectives (Wooldridge & Jennings, 1995). In this context, autonomy can be defined as the agent’s ability to act without the direct intervention of humans (or other agents), and to have control over its own actions and internal state (Nwana, 1990). When adopting an agent-oriented view, it becomes apparent that most problems require multiple agents to represent the decentralized nature of the problem, the multiple loci of control, the multiple perspectives or the competing interests. Moreover, the agents will need to interact with one another either to achieve their individual objectives or to manage the dependencies that arise from being situated in a common environment (Jennings, 2001). The need of such organization for a cooperative working set-up ensued to the development of Multi Agent System and formalization of a goal oriented communication among them.

Multiagent Systems (MAS) deals with behavior management in collections of several independent entities, or agents (Stone & Veloso, 1997). It is a subfield of AI that aims to provide principles for construction of complex systems involving multiple agents, and mechanisms for coordination of independent agents’ behaviors. In general, the agent in a single-agent system models itself, the environment, and interactions with other agents, where each agent is an independent entity with its own goals, actions, and knowledge (Stone & Veloso, 1997).

![Figure 2-6: Nwana’s typology of Agents](image-url)
Multiagent systems differ from single-agent systems as the former incorporates several agents which model each other’s goals and actions through direct interactions (using communication language). According to one of the most accepted agents’ classifications (Nwana, 1990), the agents can be categorized based on the three basic attributes: autonomy, learning and cooperation. Autonomy refers to the principle that agents can operate by themselves without the need of human guidance.

The cooperation attribute requires multiple agents, in contrast to the other attributes. In order to cooperate, the agents need to possess social ability to interact with other agents or possibly with humans through some communication language (Wooldridge & Jennings, 1995).

The third attribute is the ability to learn over a period of time resulting in the increased performance. It is essentially a feature which an agent must possess to be termed as intelligent. Figure 2-6 illustrates these three attributes, combination of which evolves four types of agents (Nwana, 1990): Collaborative agents, Interface agents, Collaborative learning agents and smart agents. According to this typology our system has been designed based on the collaborative agents.

A system such as the one presented in the thesis works best in an agent oriented paradigm which incorporates multi-agents. The reasons mainly for this are the system’s requirement of independent and yet cooperating behavior of all the components. The other benefit relates to the expansion of the system. The system designed in an agent oriented paradigm can be expanded without making much change in the existing system.

### 2.5.1 JADE for Multi Agent Communication

JADE (Java Agent DEvelopment Framework) is a software development framework aimed at developing multi-agent systems and applications conforming to FIPA\(^22\) (Foundation for Intelligent Physical Agents) standards for intelligent agents (Bellifemine, Caire, Trucco, & Rimassa, 2010). JADE is written in Java language and is made of various Java packages. It includes:

- **A runtime environment** where JADE agents can “live”. It must be active on a

\(^{22}\) [http://www.fipa.org/](http://www.fipa.org/)
given host before one or more agents can be executed on that host.

- **A library of classes** that programmers can use (directly or by specializing them) to develop their agents.
- **A suite of graphical tools** that allows administrating and monitoring the activity of running agents.

There are several multi-agent tools available contemporary to JADE, such as 3APL/2APL\(^ {23}\), Goal\(^ {24}\) Agent Programming Language, Jack\(^ {25}\), Jadex\(^ {26}\) and Jason\(^ {27}\) works on BDI (Belief, Desire and Goal) concept. IMPACT\(^ {28}\) project has its objectives to develop powerful collaborative agents in heterogeneous and distributive environment.

Though Jade has many features that enable a developer to implement a multi-agent system with interactive GUI (jade.gui) and several other packages for easy management, we preferred our own interface and therefore used jade.wrapper package. It provides JADE higher-level functionalities that allows the usage of JADE as a library. This package enabled us to connect our JSP programs based application to the JADE based agents and the container.

### 2.5.2 Accessing JADE using JSP and NetBeans

Java Server Pages (JSP) technology allows developers to build dynamic Web pages to represent information which are platform independent. JSP technology separates the user interface from content generation, enabling designers to change the overall page layout without changing the underlying dynamic content. We used NetBeans\(^ {29}\), a powerful IDE to manage the project and organizing various libraries for easy connectivity and accessing.

The JADE agents can be accessed and managed from JSP programs using Jade Gateways. Jade Gateways allows a non jade application to issue commands to a jade based application. The package **jade.wrapper.gateway** uses two classes,
JadeGateway and GatewayAgent in order to establish communication link between JSP page and Jade agents following the steps sequence\(^3^0\) given below:

- Create an application-specific class that extends GatewayAgent, redefining its method `processCommand`.
- Initialize this JadeGateway by calling its method `init()` with the name of the class of the application-specific agent.
- Finally, in order to request the processing of a command, call the method `JadeGateway.execute(Object command)`. This method will cause the callback of the method `processCommand` of the application-specific agent. The method `execute` will return only after the method `GatewayAgent.releaseCommand(command)` has been called by the application-specific agent.

Once the jade gateway gets properly connected, it can then be used to pass on the request/data from JSP to JADE platform and vice versa.

### 2.5.3 Crawlers

The Web Crawlers act as autonomous agents that are programmed to move from one hyperlink to another on the Web based on the pre-set criteria to download the URLs and related information (Castillo, 2004). In the presented work, the designs of dedicated crawlers called focused crawlers has been proposed and implemented through agents. Chapter 4 covers a detailed study over such crawlers.

### 2.6 Web Content Retrieval for Learning Purpose

The existing systems that are built for managing an easy access to learning content, generally expect people to suggest/store useful hyperlinks (URLs)/ resources under different subject domains. These links or resources are then organized by the system, to later make it available for learners as per their needs. Such systems usually depend on the community’s interaction/input to the system for the repository updates. The Web on the other hand, which acts as a huge repository of learning resources is a common platform for all communities’ interaction, which makes content on the Web

dynamic. This very nature of the Web can be utilized by extracting informal information to augment a system’s repository automatically (possibly through agents) irrespective of the users communities’ direct input. In this section we will discuss about the dynamic and useful learning resources as an alternate to the existing static repositories. Consequently, for an effective usage of these dynamic Web resources, an introduction to the Concept Ontology has been proposed.

### 2.6.1 Learning Objects as Static Repositories

Since the invention of Semantic Web in 2000, many researchers have proposed various architectures and frameworks that use LOM (Learning Object Metadata) (LOM, 2002) technology to enhance the learning experience of learners (Brank, Grobelnik, & Mladenić, 2005). Most of these frameworks initially started with the use of metadata. Efforts were made to convert e-Learning courses to metadata on a large scale (Brase & Nejdl, 2003) to facilitate a vast group of learners. Standards were established to generalize the structure for such metadata as Dublin Core and LOM. However, there were issues in using LOM. They represent static repositories and suffer from various shortcomings (Zouaq & Nkambou, 2008).

![Modular content hierarchy](image)

**Figure 2-7: Modular content hierarchy**

31 The Dublin Core Metadata Initiative,  [http://dublincore.org](http://dublincore.org)
In literature various metadata schemes have been proposed which produces reasonable results but, on the expense of creating new learning material customized to the needs of an application. SCORM Content Aggregation Model (SCORM, 2009), (Dodds, 2002) and CISCO RLO/RIO Model (Barrit, Lewis, & Wieseler, 1999) are a few of the prominent examples of Learning Object Models that identify learning object components and their use. These models by and large generalize the content creation process (Figure 2-7) which starts at raw elements and assembles or constitutes these raw elements to information objects that are meant to serve specific objective or problem.

The terms like Learning Object, Learning Object Metadata and Learning repositories are frequently referred by researchers of this area to represent learning content or resources. A much simplified description of these terms as given by Friesen (2003) is as follows. Learning Object refers to digital educational resources; metadata refers to their systematic description to facilitate searching and administration; and repositories represents online, searchable collections of these resources. More specifically, Learning Object Metadata (LOM) standard provided by IEEE Learning Technology Standards Committee defines a Learning Object as: “…any entity, digital or non-digital, that may be used for learning, education or training.” (LOM, 2002). However, this definition allows a wide verity of granularities, which means, Learning Object (LO) to include everything from a few byte content to millions of pages of content. Besides, there exist some complexities (Friesen, 2003) over the term Learning Object as well. For example the term ‘Object’ in ‘Learning Object’ refers to a specific technological paradigm called object oriented which consists of specific properties such as polymorphism, concurrency, encapsulation etc. which in contrast, has not been considered by any of the LOM based content models. Similarly learning has no clear description as to how the learning occurs or how it can be best understood. Another problem lies with the exchange of content or information among many models that are developed independently with huge investments. Moreover, the architecture requires application based customization. This means one has to put more efforts in developing a complete system for separate institutes, organizations or for other purposes. Even the use of same model with different purpose or institutes requires lot of initial efforts to actually deploy it. Thus, it is hard
to provide desired results, unless the whole structure is automatically implemented and semantically linked.

Duval and Hodgins (2003) suggests Learning Object taxonomy which identifies different kinds of LOs and their component to overcome vagueness in the definition. They discussed sixteen Research issues on Learning Objects and their use in education and training. The work in thesis focuses on the first two research issues among them, specifically for designing the repository structure.

The first research issue defines the Learning Object Taxonomy to redefine the basic definition of IEEE LOM Learning Object in view to the vagueness in LO definition. The taxonomy in Figure 2-7 shows different levels from domain independent components to more specific application dependent collections. At the most basic level of granularity there exist Raw Data Media Elements, which are contained within the Information Objects. Learning objects contain Information Objects. Aggregate Assemblies contain LOs with some defined objective. This is further assembled into larger collections, like courses and whole curricula.

The second research issue deals with the LO component Architecture and suggests the need to develop a flexible architecture that enables:

- Structuring of LOs and their components that separates content, structure and presentation.
- Interaction between LOs and their components.

These research issues helped in providing a clear vision to work for a quality system.

2.6.2 Web Content as Dynamic and Useful Learning Resource

The biggest hurdle in any content learning management system is the content creation. The content in the system needs frequent updates and improvements as the learning content changes with time due to the new discoveries/inventions. It is practically difficult to upgrade the knowledge base manually and to keep accordance with the fast pace of changes.

On the contrary, the Web allows participation of a large and diverse group of people across the globe. This leads to a huge content creation with varied level of
descriptions of the content and sources with continuous changes. The Web is dynamic and full of useful learning resources. The need is only to device some automatic mechanism to capture this information and organize it in a knowledge base for a learner. This mechanism can work automatically if the knowledge base is represented with a language that is understandable by humans and machines as well. With this intent we propose a structure represented in Web Ontology Language, called Concept Ontology.

2.6.3 Concept Ontology for Relevant Web Resource Retrieval

The ontology layer in the Semantic Web architecture (Figure 2-3) is identified as a standard layer of the knowledge based systems (Neches, et al., 1991). This ontology having a language OWL understandable by both humans and machines enables us to build a system that captures the relevant information from the Web through software agents. The proposed Concept Ontology is an ontology specially constructed to support the organization of concepts under various academic domains. This ontology is consumed by the software agents to retrieve relevant information from the Web. The agents are able to determine the relevance as they can read and understand the Concept Ontology. Not limiting to this, the agents are also able to augment the Concept Ontology with the gathered information from the Web, by using knowledge derived from the Concept Ontology. The next chapter explains a proposed design for the Concept Ontology to serve above purposes.