SEARCH SYSTEM FOR ONTOLOGY

THESIS

submitted in fulfillment of the requirement of the degree of

DOCTOR OF PHILOSOPHY

to

YMCA UNIVERSITY OF SCIENCE & TECHNOLOGY

by

VANDANA DUA

Registration No. YMCAUST/Ph12/2010

Under the Supervision of

Dr. KOMAL KUMAR BHATIA

PROFESSOR

Department of Computer Engineering

Faculty of Engineering and Technology

YMCA University of Science & Technology

Sector-6, Mathura Road, Faridabad, Haryana, India

JUNE, 2016

DEDICATED

to

My Family

DECLARATION

I hereby declare that this thesis entitled **SEARCH SYSTEM FOR ONTOLOGY** *by* **VANDANA DUA,** being submitted in fulfillment of the requirements for the Degree of Doctor of Philosophy in COMPUTER ENGINEERING under Faculty of Engineering and Technology of YMCA University of Science & Technology Faridabad, during the academic year 2016, is a bonafide record of my original work carried out under guidance and supervision of **Dr. KOMAL KUMAR BHATIA, PROFESSOR, DEPARTMENT OF COMPUTER ENGINEERING** and has not been presented elsewhere.

I further declare that the thesis does not contain any part of any work which has been submitted for the award of any degree either in this university or in any other university.

(Vandana Dua)

Registration No. YMCAUST/Ph12/2010

CERTIFICATE

This is to certify that this thesis entitled **SEARCH SYSTEM FOR ONTOLOGY** by **VANDANA DUA,** submitted in fulfillment of the requirement for the Degree of Doctor of Philosophy in **COMPUTER ENGINEERING** under Faculty of Engineering & Technology of YMCA University of Science & Technology Faridabad, during the academic year 2016, is a bonafide record of work carried out under my guidance and supervision.

I further declare that to the best of my knowledge, the thesis does not contain any part of any work which has been submitted for the award of any degree either in this university or in any other university.

(Signature of Supervisor)

Dr. Komal Kumar Bhatia

PROFESSOR

Department of Computer Engineering

Faculty of Engineering and Technology

YMCA University of Science & Technology, Faridabad.

Dated:

ACKNOWLEDGEMENT

Foremost, I want to offer this endeavour to GOD for the strength, peace of my mind which the Almighty has bestowed upon me.

I express my sincere gratitude to Honourable Guide **Dr. Komal Kumar Bhatia** for giving me the opportunity to in this area. It would have never become possible for me to take this thesis to this level without his innovative ideas and his relentless support and encouragement. It is worth to mention the encouragement and initiative sincerely rendered by **Dr. A.K. Sharma.** My special thanks to **Dr. Naresh Chauhan**, Chairman, YMCAUST for his wishes and kind support. I am thankful for the valuable suggestions provided by **Dr. Nagpal, Dr. Manjeet, Dr. Jyoti and Dr. Neelam** It's my pleasure to offer my thanks to staff at **YMCA**, all my friends, whose good wishes always remained with me in completing my work.

I would like to thank my HOD, **Prof. S.I Patil** and Dean, **Prof. Aditya Abhyankar** for their constant encouragement, motivation and support for completing my PhD work. I take this opportunity to thank **Dr. Vishal Naranje** for the valuable discussions and moral support. I am especially thankful to **Dr. Sujatha Raman**, for all the suggestions, feedback and support in the compilation of thesis. I wish to thank **Mr. Santosh Lolage** for his constant motivation and support. I wish to offer my thanks to all the faculty members, staff members of **Department of Technology, SavitriBai Phule Pune University.**

Words are not enough to express my gratitude to my dear kids **Naina** and **Pulkit,** my husband **Chander Mohan Dhingra** for their patience, compromises, and support. I am thankful to my mother-in-law **Smt. Darshan Kumari**, and father-in-law **Sh. Hem Raj Dhingra** for their constant support and love. My greatest gratitude goes to my father **Sh. Amar Nath Dua** who has remained with me at each and every step of my journey and mother **Smt. Bimla Dua** for her support and wishes which gave me the strength to complete the work. I would like to thank my sister-in-law and her husband **Dr. Parveen Batra, Dr. Vinod Batra** and my brothers and sisters-in-law **Aman Dua, Kriti Dua, Nitin Dua and Alisha Dua** for their love, encouragement and support for completing my PhD.

(Vandana Dua)
Registration No.- YMCAUST/Ph12/2010

ABSTRACT

World Wide Web (WWW) is a huge repository of information and its success is due to its decentralized structure where anyone irrespective of its geographic location can publish its content. However, due to large amount of available information it is becoming difficult to access the relevant information. There has been a lot of paradigm shifts in the searching of information from WWW, which varies from directories to search engines. Web search techniques are mainly based upon a combination of textual keywords in association with ranking of the documents depending on the link structure of the web. Such keyword based search engines are the main tools for retrieval of information from WWW. However, the main limitation of keyword based approaches is that it produces irrelevant and imprecise results, making the search ineffective and time consuming, thus mandates an approach towards Semantic Web.

Semantic Web allows the machine to interpret the concepts with underlying knowledge representation in structured languages such as RDF, OWL. The main technologies for the Semantic Web are ontologies, metadata and the semantic web language RDF, OWL, DAML+OIL. The current search engines cannot effectively utilize semantic web documents for web search as current search engines are not able to exploit the information representation provisions available in languages of semantic web. Therefore, the traditional techniques cannot be deployed for searching the documents represented in semantic web languages, hence there arises a need for modified crawling; indexing and ranking mechanism as the existing mechanism are inadequate for it.

The present thesis work makes a contribution to the research effort of designing and developing of a framework for searching ontology annotated documents. Three ontologies have been developed in the domain of laptop using Protégé tool. The developed ontologies are associated with the web pages that best describes the concepts contained in particular ontology. Framework for crawling these ontology annotated web documents: SemCrawl, framework for indexing: SemIndex, framework for Ranking: SemRank has been proposed and implemented. The

developed framework SemEngine returns the web documents based on the concept coverage of the ontology depending on the query entered by the user.

The ontologies were developed in Protégé tool successfully and have been validated for consistency with the set of queries using Description Logic. The developed system is efficient in terms of index creation as the index size created requires less space in comparison to index created for whole web page. The developed SemRank system for ranking ontology annotated web pages has been validated using Pearson Correlation coefficient which gives high collinear correlation. The system has been compared with the traditional mechanism of searching using the evaluation metrics that shows an improvement over the existing system. The developed system is capable of downloading the semantic web documents and effectively creates the index and ranks the web pages giving the results with good relevance.

LIST OF TABLES

Table	Title	Page No.
Table 2.1	Building an Inverted Index-Step1, 2, 3	15
Table 2.2	Building an Inverted Index-Step5	16
Table 2.3	Difference between Traditional Web and Semantic Web	19
Table 2.4	Some Constructs of XOL language	28
Table 2.5	Different tags of SHOE	30
Table 2.6	Language Constructs of DAML	35
Table 2.7	Comparison of different Ontology Languages	38
Table 2.8	Comparative Analysis of ontology editing tools	45
Table 2.9	Weights of Hyperlinks	50
Table 2.10	Summarization of various Ontology Tools	63
Table 3.1	Comparative analysis of Search Engines	67
Table 3.2	Index of <Subject, Predicate, Object, Ontology>	73
Table 3.3	Index of <HTML file, Ontology>	74
Table 4.1	Domain and Range of Slots	89
Table 4.2	Instances for *University* Ontology	89
Table 4.3	Performance Rates for different Search Queries	91
Table 4.4	Classes for Laptop_Review Ontology	92
Table 4.5	Instances for Laptop_Review	93
Table 4.6	Classes and SubClass of Laptop_Specification Ontology	94
Table 4.7	Instances for Laptop_Specification Ontology	94
Table 4.8	Classes and Subclasses of Laptop_Seller ontology	97
Table 4.9	Set of queries to be executed in the domain of laptop using three ontologies.	99
Table 5.1	Series of Test conducted on Repositories for Crawler Module	111
Table 6.1	Comparative analysis of indexing Techniques	117
Table 6.2	HTML File and the associated Ontology	124
Table 7.1	Comparative analysis of ontology ranking algorithms	128
Table 7.2	Query concept for Laptop Cost	136
Table 7.3	Concept_Weight_Measure for ontologies	136
Table 7.4	Measure for Relatedness	137

Table	Title	Page No.
Table 7.5	Measure for Match-Measure	137
Table 7.6	Overall computed Rank_Score	137
Table 7.7	Rank Score for Web pages	138
Table 7.8	Ranked results for Query “laptop display”	138
Table 7.9	Human Rank vs. Proposed Approach	138
Table 8.1	Web Pages with the associated ontology	144
Table 8.2	Set of evaluation query	147
Table 8.3	Experimental Evaluation of the System	150
Table 8.4	Comparative analysis of Search Engines	154

LIST OF FIGURES

Figure	Title	Page No.
Figure 1.1	Depiction of flow of outline of the Thesis	7
Figure 2.1	Architecture of Search Engine	11
Figure 2.2	Basic Architecture of a Crawler	12
Figure 2.3	Algorithm for Crawler	13
Figure 2.4	Hub and Authority nodes	17
Figure 2.5	Semantic Web Layered Architecture	21
Figure 2.6	Example of XML describing a web page	22
Figure 2.7	Example of Logic	23
Figure 2.8	Languages Stack in Semantic Web	28
Figure 2.9	An example of XOL Language	29
Figure 2.10	Example of SHOE Ontology for CS Department	30
Figure 2.11	Example of OML	32
Figure 2.12	Dividing the RDF statement into <S, P, O>	33
Figure 2.13	RDF Graph Representation of statement	33
Figure 2.14	Example of OIL defining classes	34
Figure 2.15	Example of DAML	36
Figure 2.16	University ontology specified in OWL language	37
Figure 2.17	Protégé Architecture	39
Figure 2.18	GUI Interface for Protégé	40
Figure 2.19	Architecture of WebODE	41
Figure 2.20	Architecture of NeOn Toolkit	43
Figure 2.21	Swoop Architecture	44
Figure 2.22	SWRL Rules implemented in Protégé	46
Figure 2.23	Example of Rule Language	48
Figure 2.24	Hyper linking of RDF fragments	49
Figure 2.25	Architecture of Swoogle	52
Figure 2.26	Architecture of Falcon Search Engine	55
Figure 2.27	Flow of Ontosearch	58
Figure 2.28	SquishQL Query	59
Figure 2.29	Example of SPARQL	61

Figure 2.30	Example of SQWRL	62
Figure 3.1	Google result for query “Laptop Specification”	66
Figure 3.2	Proposed Architecture for Search System for Ontology Annotated Web Documents	70
Figure 3.3	Pseudo code for SemCrawl Module	72
Figure 3.4	Parsing of results to <HTML, Ontology>	74
Figure 3.5	Ontology and the web page associated with that ontology	74
Figure 3.6	Algorithm for SemIndex Module	75
Figure 3.7	Concept matching of Query	77
Figure 3.8	Pseudo code for Search Module	78
Figure 3.9	Three ontologies developed in Laptop domain	78
Figure 3.10	Methodology of developed system	79
Figure 3.11	Functional Diagram of the developed system	82
Figure 4.1	Class Hierarchy for *University* Ontology	87
Figure 4.2	Google result for Query “laptop price range 20000 to 300000”	90
Figure 4.3	OntoVisualizer Result of Laptop_Review Ontology	92
Figure 4.4	Laptop_Review Ontology Class	93
Figure 4.5	Laptop_Specification Ontology	95
Figure 4.6	Ontovisualizer result of Laptop_Specification Ontology	96
Figure 4.7	Laptop_Seller Ontology	97
Figure 4.8	Ontovisualizer result of Laptop_Seller Ontology	98
Figure 4.9	Execution of Query for Laptop_Specification Ontology	100
Figure 5.1	Architecture of Proposed SemCrawl	104
Figure 5.2	Pseudo code for Fetch Module	105
Figure 5.3	Pseudo code for Filter Module	105
Figure 5.4	Pseudo code for Link Extraction Module	106
Figure 5.5	Structure of a URI Dispatcher Module	107
Figure 5.6	Pseudo code for Parser Module	107
Figure 5.7	<S, P, O> Triples	108
Figure 5.8	Methodology for development of SemCrawl Architecture	108
Figure 5.9	Repositories of web pages	110
Figure 5.10	Filtered web pages which are annotated with ontology	111
Figure 5.11	Extraction of Subject, Predicate, Object from Ontology	112
Figure 5.12	Console output of Extracted Subject, Predicate, Object	113

Figure 6.1 RDF/XML Code 114
Figure 6.2 Triple representation of RDF/XML Document 115
Figure 6.3 Graph Representation for RDF/XML document 116
Figure 6.4 Architecture of SemIndex for indexing crawled Ontologies 119
Figure 6.5 Pseudo code for Indexer Module 120
Figure 6.6 Instances of Laptop_Seller Ontology 120
Figure 6.7 Example of a Triple <Subject, Predicate, Object> 121
Figure 6.8 Triple representation 121
Figure 6.9 Algorithm for Ontology extractor 121
Figure 6.10 Example <HTML File, Ontology> index 122
Figure 6.11 Index of<Subject, Predicate, Object, Ontology> 122
Figure 6.12 Index of < HTML File,Ontology> 123
Figure 6.13 Parsing of results to <S, P, O> 123
Figure 6.14 Ontology Extractor Output 124
Figure 6.15 Ontology index of <Subject, Object, Predicate, Ontology> 125
Figure 6.16 Index of < HTML file, Ontology, > in database 125
Figure 6.17 Graph of HTML web contents without Ontology vs using SemIndex126
Figure 7.1 Proposed Framework for Ranking Ontology Annotated Web Pages 131
Figure 7.2 Algorithm for Match-Logic (O) of Ontology 133
Figure 7.3 Algorithm for Annotated Web Page Extractor 134
Figure 7.4 Algorithm for Rank Logic Module 135
Figure 7.5 Human approach vs. proposed approach 139
Figure 8.1 Development of Laptop_Specification Ontology 143
Figure 8.2 Crawler Module Result 145
Figure 8.3 Ontology Parsing Result 146
Figure 8.4 Ontology Index 146
Figure 8.5 Execution of Query in Protégé 147
Figure 8.6 SemEngine Result Output 148
Figure 8.7 SemEngine result for query “laptop seller sites” 149
Figure 8.8 SemEngine Precision Curve vs Google 151
Figure 8.9 SemEngine Curve of Ideal vs Actual Recall 151
Figure 8.10 SemEngine F-Measure Ideal vs Actual Curve 152
Figure 8.11 Precision Graph for the increased number of queries 153

Figure 8.12 Recall Graph for the increased number of queries 153
Figure 8.13 F-measure graph for the increased number of queries 154

LIST OF ABBREVIATIONS

DL	Description Logic
DAML	Darpa Agent Markup Language
FOAF	Friend-of-a-friend
HITS	Hypertext Induced Topic Search
HTML	Hypertext markup Language
HTTP	Hypertext transfer Protocol
RDF	Resource Description Framework
RDFS	Resource Description Framework Schema
OIL	Ontology Interface Layer
OML	Ontology Markup Language
OWL	Web Ontology Language
RuleML	Rule Markup language
SPARQL	SPARQL Protocol and RDF Query language
<S,P,O>	<Subject, Predicate, Object>
SHOE	Simple HTML Ontology Extension
SWO	Semantic Web Ontology
SWDB	Semantic Web Databases
SWRL	Semantic web rule Language
SquishQL	SQL like Query language
TF-IDF	Term Frequency-Inverse Document Frequency
W3C	World Wide Consortium
XML	Extensible Markup Language
XOL	An XML-Based Ontology Exchange Language
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
UI	User Interface

BRIEF PROFILE OF THE RESEARCH SCHOLAR

Name: Ms. Vandana Dhingra

Designation: Assistant Professor, Department of Technology,
SavitriBai Phule Pune University, Pune

Qualification: Ph.D (2010- to current)
M.Tech (2006), First Class with Distinction
B.E (1998), First Class with Distinction

Research Interests: Semantic Web, Information Retrieval, Ontology, Web Mining.

Work experience:

- Assistant Professor, Department of Technology, S P Pune University, Pune (2013-to current)
- Assistant Professor, Apeejay Stya University, Gurgaon (2010-2013)
- Assistant Professor & HOD, Apeejay College of Engineering, Gurgaon (2004-2010)
- Lecturer, CITM College of Engineering, Faridabad (2002-2004)
- Lecturer, Vaish College of Engineering, Rohtak (1999-2001)

CHAPTER I

1. INTRODUCTION

1.1. GENERAL

World Wide Web (WWW) [1] is a huge source of interlinked documents that forms a very useful information source. The success of WWW is largely due to its decentralized design structure [2] where the information is hosted by several servers, and a document can point to other documents irrespective of its geographic location. An information retrieval [3, 4] is a technique for searching the information about a subject over enormous number of resources relevant to the user's information need. Information retrieval can be precisely defined as:

"Information Retrieval (IR) is finding material (usually documents) of an unstructured nature (usually text) that satisfies an information need from within large collections (usually stored on computers)" [4].

WWW has revolutionized the means of data availability. But due to its current structure [5] it's getting difficult to access the relevant information from such a large collection. The Web size has grown to a large extent and due to large volume of available information; it is becoming difficult to locate useful information [2, 6]. Retrieving the relevant information from WWW is an unprecedentedly a difficult task.

With such a large collection of information, search engines [7] are emerging as an important tool for searching the relevant information. The information is searched through search engine by submitting queries that are in the form of keywords and as a result information seekers find the required information. Thus, search engines are considered as an important tool for information retrieval system that returns a set of ranked web pages according to their relevance and matches the query keywords.

1.2 SEARCH ENGINES

Search Engine is a tool that is used to retrieve the information stored over the WWW. Typically Search Engine has the following main components:

1.2.1 Crawling

It is the first stage of search engine in which the documents from the web are downloaded based on the URL received from the URL Frontier Queue [8]. The web pages fetched from the web are sent for parsing, for further extraction of links. The extracted links are sent to URL Frontier Queuefor fetching of web pages from those links after passing through a series of test of duplicate contents and URL elimination.

1.2.2 Indexing

The crawled web pages are then indexed by the Indexer Module. The major steps involved in index construction are -Tokenization, linguistic pre-processing process such as hyphenation, stop word removal, stemming, lemmatization, normalization [4]. These terms are sorted and maintained as posting list consisting of the frequency of the terms and the document that each term occurs in.Different type of indexes are constructed depending upon the type of contents; Text Index, Structure Index,Utility Index [7].

1.2.3 Searching

Query terms entered by the user are compared with the index, producing the results. When a user query is entered, the terms of query are matched with the terms in the index structure and the terms matching the query terms are returned as a result to the user.

1.2.4 Ranking

The web pages returned after matching with query are ranked based on various factors. The most widely used ranking algorithms are Page-Rank and Hyper-text Induced Topic Specific (HITS) algorithm.

The search engines, for example Google, Yahoo etc. match the keywords in the query with the web pages that are having those keywords, resulting into result page set which has relevant and irrelevant results. Retrieving the relevant information from the information available is an important research issue in search engines.

1.3 LIMITATIONS OF THE TRADITIONAL SEARCH ENGINES

Major search engines such as Google, Yahoo works on keyword based matching [9]. It is the user's work to extract out the relevant information from a large set of results. Finding out the relevant information from such a large set of web pages proves out to be very tedious task. Search engines based on keyword matching have certain problems associated with them [10, 11, 12, 13, 14] as listed below:

1. High recall, low precision

The main issue with the returned results are that they have high recall but low precision which means that it returns a lot of important results from its repository but those results are not that relevant which refers to low precision. But with lot of results retrieved is that even if the main relevant pages are retrieved, they are of little use if large numbers of mildly relevant or irrelevant documents are also retrieved.

2. Low or no Recall

Often it happens that users don't get any relevant answer for request, or important and relevant pages are not retrieved.

3. Lack of machine Understandability

The machine has the inability to understand the provided information due to lack of universal format [15]. The information is based on HTML based free format web pages which are very suitable for direct human use but is not appropriate for the automated information exchange, retrieval and processing by software agents(machines). The current web contents are mostly represented in HTML which is more presentation language and henceforth, does not help in machine interpretability.

4. Poor Content Aggregation

For the query entered the results are lot of documents or web pages; a user has to manually aggregate the partial information to get the complete information. Hence, search engines returns a lot of results which has to be manually aggregated.

5. No Semantics

Results are based on just matching of the keyword in that document. There is no concept based matching of the query with the documents. Therefore, the results may or may not be relevant in context of semantic to the user query.

6. Difficulty in handling queries with disambiguous terms.

The current search engines matches the query keywords with the keywords present in the document. For example query "jaguar" has two different meaning car as well as animal and hence, produces results for both the documents, leading to low precision. Similarly, the query "holiday "and "vacation" relates to the same term but when entered separately produces different set of results although referring to same word.

The limitations specified above mentions that just matching keywords do not help in searching; it produces a lot of imprecise results. The efficient searching requires the machine to understand the semantics of the information. This machine understandability concept can help WWW to make a move from syntactic web [16] to Semantic Web [16, 17].

1.4 MOTIVATION OF THE RESEARCH

Web search is a key technology of the Web, since it is the primary way to access content over the WWW. Current web search is essentially based on a combination of textual keyword search with an important ranking of the documents depending on the link structure of the web [18]. The current web is based on HTML, [19] which specifies how to layout a web page for human readers. HTML as such cannot be exploited by information retrieval techniques to improve results, which has thus to rely on the words that forms the content of the page; hence it is restricted to keywords [20]. But as discussed above the limitations of keyword based search leads to movement towards Semantic Web. In Semantic Web, the knowledge is represented

with ontologies which can be specified in highly structured languages such as RDF, OWL [21]. These representation languages have high expressive constructs.

There has been a lot of research in the area of Semantic Web. For example, Swoogle [22] search system provides a search for Semantic Web documents and terms. The output system of the Swoogle represents a set of ontologies represented in Semantic Web languages [23]. These ontologies can be used to annotate web pages, which results in concept based searching of the web pages.

With the advancement in the area of Semantic Web, the main work focuses on the knowledge representation of these web pages. But only a few research efforts have been found where these ontologies can be used to represent web pages for information retrieval. Therefore there is a requirement to use the web page which are represented with knowledge based approaches and a system need to be developed to crawl, index, rank and search those web pages. .

1.5 OBJECTIVES OF THE PROPOSED WORK

The overall goal of the proposed work is to develop a search system for concept based searching. The specific objectives of the present work are as follows:

1. To develop ontology in a particular domain.

Ontology need to be developed using development framework which covers domain knowledge.

2. To develop a crawler system to crawl these ontologies and annotated web pages.

A crawling mechanism need to be developed for crawling the web pages which are annotated with ontology, traditional crawlers cannot be reused for this as semantics are associated with the web pages under research which cannot be crawled by traditional crawling system.

3. To develop an Indexer system to index the web pages and the corresponding ontology.

An indexing mechanism need to be developed for indexing web pages annotated with ontology. Different indexing mechanism is required to be deployed as web pages are represented with semantics which are parsed as subject, predicate, object.

4. To develop a ranking system to present ranked retrieval system.

A ranking system need to be developed to rank the ontologies and the associated web pages. This ranking mechanism needs two different approaches first to tank ontologies and then to rank web pages.

5. To develop a search system to searching these ontology annotated web pages.

A search system needs to be developed for the proposed algorithms which will search the web pages based on the concepts of the query.

To achieve these objectives domain specific approach need to be followed to develop ontology in a particular domain. A concept based crawling, indexing and ranking system need to be developed to crawl these ontologies and annotated web pages. In order to assure the practical implications of the objectives undertaken, the system should support scalability, extensibility, robustness and improvement over the evaluation metrics. A search system need to be developed which can search the web pages based on the concepts associated with the pages and should have high performance.

1.6 ORGANIZATION OF THESIS

The present thesis is organized in nine chapters and is as shown in Figure 1.1

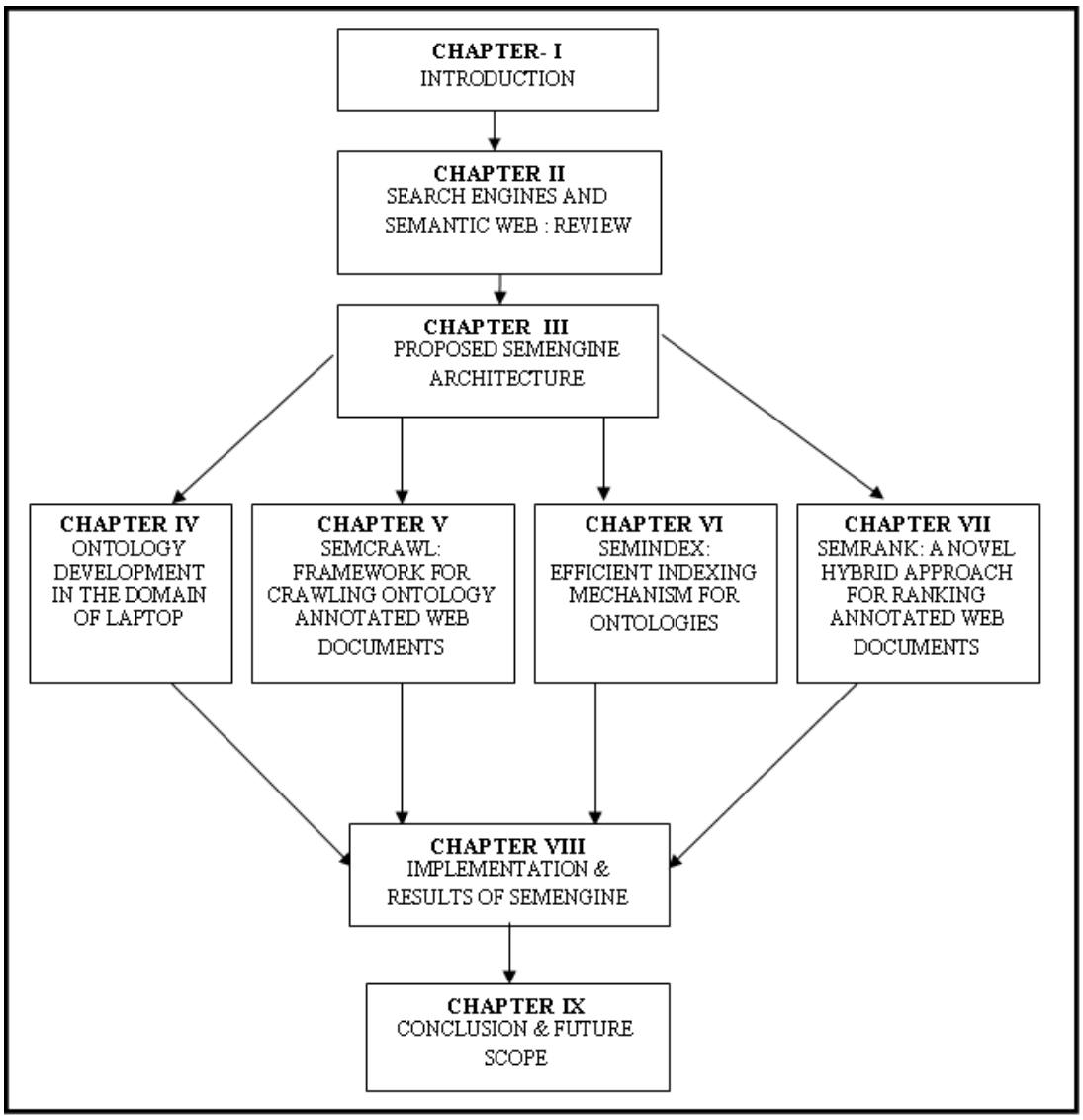

Figure 1.1 Depiction of flow of outline of the Thesis

The content of each chapter for thesis is summarized as under:

- **Chapter II** reviews the published work related to search engines and Semantic Web. The chapter discusses the search engines, limitations of the current search engines and need for semantic Web. Research work carried out in the area of Semantic Web, ontologies has also been discussed in detail.

- **Chapter III** presents a comparative analysis of various search engines and proposes the architecture of novel search engine. The phases of development of proposed SemEngine have been discussed in detail. The methodology formulated for the research has been discussed. The implementation of the proposed SemEngine architecture has been discussed in this chapter with the results.

- **Chapter IV** discusses the ontology development in a particular domain. The problem has been discussed and the related ontologies have been developed in Protégé.

- **Chapter V** proposes the architecture for crawling semantic annotated web pages. The architecture of proposed SemCrawl has the feature of extracting semantically annotated web pages. This chapter describes the detailed architecture of SemCrawl and all its components.

- **Chapter VI** proposes the architecture for indexing semantic annotated web pages. The architecture of proposed SemIndex creates an index for an ontology which is related to a web page and conceptual index of ontology. This chapter discusses the architecture of the proposed SemIndex and its component in detail.

- **Chapter VII** proposes the architecture for ranking semantic annotated web pages retrieved as a result of user query. The proposed architecture SemRank ranks the web pages in the order of their relevancy based on various factors.

- **Chapter VIII** discusses the implementation details of the modules created and the results of SemEngine architecture.

- **Chapter IX** presents the contributions of the present research and suggestions for future research.

Chapter II

2. SEARCH ENGINES AND SEMANTIC WEB: REVIEW

2.1 INTRODUCTION

This chapter presents the literature survey covered in nine sections. First section introduces the World Wide Web, the architecture of search engine which is used for retrieval of information from the web, the problem with the current information retrieval model and differences between the traditional web and the Semantic Web. Second section details about the layered Semantic Web architecture. Third section discusses about the technologies for Semantic Web. Forth section provides the details of ontology, like different definitions given for ontology, reasons for developing ontology, ontology languages. Fifth section presents about the different ontology development tools. Sixth section of this chapter discusses about the different ontology rule languages. Seventh section enlists various semantic search engines and their architecture. Eighth section discusses about the Semantic Web query languages and ninth section narrates about the summary of various ontology tools.

2.1.1 Evolution of World Wide Web

Internet [24] is collection of large number of interconnected computers distributed across different geographical location over the world. The evolution of the internet started with the Advanced Research project Agency (ARPA) [25] project started by the US Department of Defence for military purpose called as ARPANET in 1960s to transmit the data using circuit switching. In 1969, ARPAnet was linked as a research project with four universities -University of California Los Angles (UCLA), University of California-Sant Barbara, Stanford University and University of Utah. With these universities, ARPAnet project developed packet switching concept which is used to send message from one remote location to another.

In 1972, when ARPANET was made publically available, electronic mail [26] was proposed by Ray Tomlinson and in 1973; TCP/IP Protocol [24] was introduced after

which Internet began to work formally. Internet is a system of interconnected computers networked through standardized communication protocols. With the advent of Internet more and more functionalities were added i.e. Telnet, DNS, Gopher.

Tim-Berner Lee in 1989 introduced the World Wide Web (WWW) [1] by proposing the linking of documents over the internet using hyperlinks. There is a difference between both terms although they appear to be same. Internet is a networking infrastructure which connects a million of computers globally whereas WWW is one of the services running on Internet which defines a way of accessing information over internet. As a part of research project, languages and protocols for World Wide Web was developed and standardized that includes the emergence of language HTML [27] for representing web documents, URLs for identifying a web page.

2.1.2 Information Retrieval from Web: Search Engine Architecture

The amount of information available over the WWW is growing rapidly and in very hysterical manner as every user is allowed to read and publish his/her contents in one or another form by using emerging technologies like blogs[28], wikis[28]. This read-write web model has made the size of the web grow with a great extent. Finding information from such a large information collection is unprecedentedly a very tough task. Search engines [30] are the tools developed for finding information from unstructured collection of information. Various Search engines were developed to access the information like Excite in 1993[31], Yahoo in 1994[32], Lycos in 1994[33], Alta Vista in 1995[34], Google in Stanford university by Sergey Brin and Larry page in 1997[30, 35] and MSN in 1999[34].

A general purpose Web Search Engine receives the query in the form of natural language from the user on user interface, processes the query, retrieves relevant documents from the index and ranks them according to their importance before supplying them to the user [38]. The main functional components of a search engine are shown in Figure 2.1.

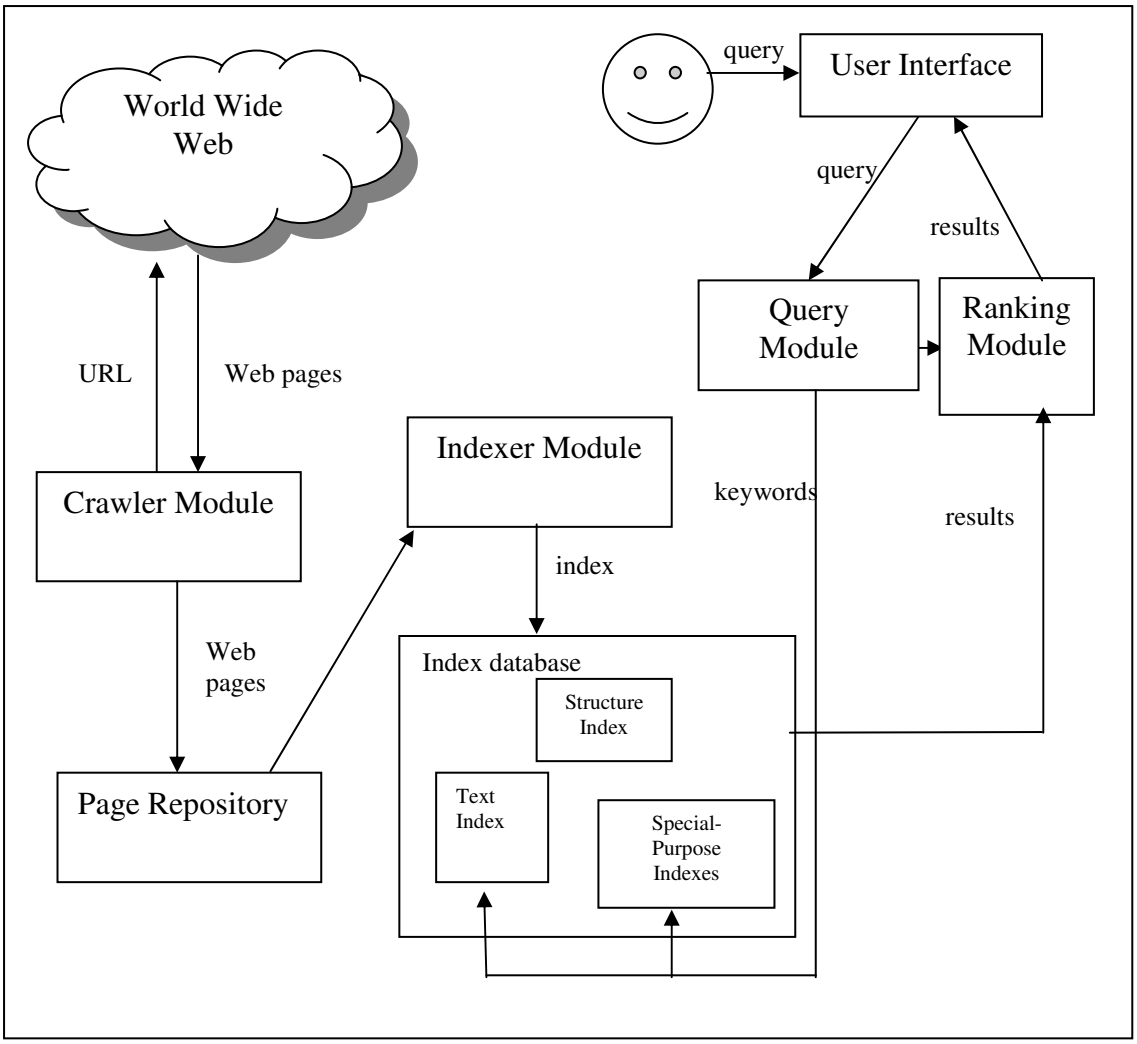

Figure 2.1 Architecture of Search Engine

The architecture of a Search Engine consists of following main components:

1. **Crawler Module**

A crawler traverses the WWW and retrieves web pages from it. The retrieved web pages are then stored in Page Repository for further indexing and are analyzed for hyperlinks contained in those pages which points to other web pages. Those hyperlinks are stored in a queue and web pages are fetched from the WWW and are finally stored in Page Repository which forms a collection of web pages downloaded from the web. The web crawlers are also called as “spiders” [9, 36, 37], “wanderers” [37]. The basic architecture of a crawler [4] is depicted in Figure 2.2

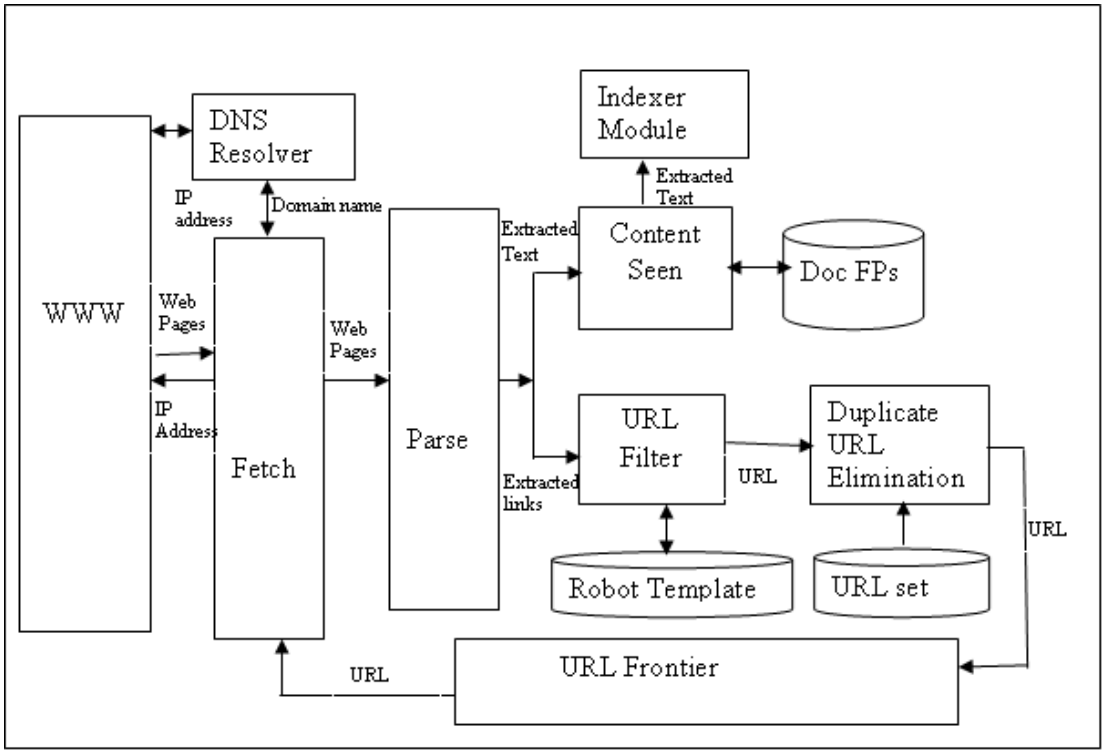

Figure 2.2 Basic Architecture of a Crawler

The basic steps for crawler module as described in Figure 2.2 are as:

STEP 1. Fetch module fetches a seed URLs from set of URLs stored in the URL Frontier and downloads the particular web page corresponding to that URL, generally using HTTP protocol. DNS Resolver module determines the IP address of web server host name of newly discovered URLs.

STEP 2. Fetched page is parsed and links and contents are extracted from that page. The extracted text and links pass through a series of test.

STEP 3. First test is done to check if a web page with the same content has already been sent to another URL by using Fingerprints (FPs) techniques with the checksum stored in Doc FPs and the duplicate contents not considered, and rest are sent to Indexer Module for indexing.

STEP 4. Second test, URL Filter is used to check whether extracted URL should be excluded from URL Frontier based on various tests. For example based on robot exclusion protocol that specifies certain URLs not to be crawled those URLs are filtered.

STEP 5. Third test is done to check for duplicate elimination of URLs.

STEP 6. After passing through a series of test, those URLs are sent to URL Frontier for further fetching of web pages from the web.

The crawler module will recursively add newer URLs to the URLs frontier and keep on fetching web pages from World Wide Web for further indexing of the documents.

The basic algorithm for crawler module is as described in Figure 2.3[39]

Crawler ( )
{
While not empty (list of URLs)
1. *read a URL from the set of seed URLs;*
2. *determine the IP address for the host name;*
3. *if robot.txt file exist on system file and includes .disallow statement then break;*
4. *determine protocol of underlying host; based on the protocol download the document;*
5. *identify for the format of file;*
6. *check whether document has been already downloaded or not*
7. *if not*
    a. *read the document and extract the links from that document else continue;*
8. *convert URLs links into their absolute URL equivalents;*
9. *add URLs to set of seed URLs;*

}

Figure 2.3 Algorithm for Crawler

Many hosts place certain access policy to crawl their contents under standard Robot exclusion Protocol which is defined as *robot.txt* file which mentions the list of pages or URLs or directories that should not be accessed from that site. Every crawler must adhere to the access policies mentioned in the *Robot.txt* file.

Example of *Robot.txt* File

*User-agent:**
Disallow: /ymcaust/circulars/
Disallow: /ymcaust/moodle/

These lines of code indicate that no robots or crawler should visit any URL starting from */ymcaust/circulars/* and */ymcaust/moodle*. Thus the crawler is able to download only those web pages that are allowed to be downloaded by the *robot.txt*.

2. Indexer Module

The contents of the crawler module are given as input to the indexer module to create various types of indexes. The different types of indexes maintained by the search engines are as follows:

1. Text index [30, 38, 40] contains the indexes created for the text which has an index for the tokens encountered in a web page to the document containing the token.
2. Structure index [38] contains the link structure of the web page.
3. Utility indexes [38] are created for the images and pdf files.

The indexing module helps in creating the indexes for the web documents in an offline mode which helps the fast retrieval of the query entered, as the keyword will be looked upon in the index data structure for fast searching. Inverted index is the most widely and generally used index for search engine. The basic steps for index construction is described below with an simple example-

An Inverted Index [4] is organized by index terms. The terms are identified from the document after initial step of pre-processing of documents which are stop word removal [4, 41], stemming word [4, 41], normalization [4, 41]. For each term identified, the index contains a list of documents which has that term. The different steps for index construction are -

Step 1: All the tokens (terms) are identified from the document and the document that contain those terms.

Step 2: The words that are common words and appear to have little value in selecting documents are removed from the vocabulary. For example is, are, the, a, such words are called stop words.

Step 3: Token normalization is done for the tokens that appear to be same by creating equivalence classes for example removing hyphens.

Step 4: Stemming and lemmatization is done to reduce inflectional and derivational forms of words to a common base form.

Step 5: The terms are sorted alphabetically with frequency of the term in document called as dictionary and has pointer to the list of documents they are contained in called as posing list.

Consider an example of two documents for building inverted index using the above described steps

Doc1: laptop sale is increasing

Doc2: all varieties of dell laptop are available for sale

Table 2.1 Building an Inverted Index-Step1, 2, 3

Term	Doc ID
Laptop	1,2
Sale	1,2
Increasing	1
varieties	2
Dell	2
available	2

Table 2.1 shows the tokens identified after Step1, 2, 3. In Step 2 stop word removal is done where is, all, of, are, for are removed. In step 3 normalization is done. In Step 4 stemming and lemmatization is done, for the current example increasing is reduced to increase, varieties reduced to variety. The inverted index constructed after step 5 is shown in Table 2.2.

Table 2.2 Building an Inverted Index-Step5

Term	Frequency	Document (Posting List)
Available	1	2
Dell	1	2
Increasing	1	1
Laptop	2	1,2
Sale	2	1,2
Varieties	1	2

The above example describes the steps for Index construction. The index construction involves identifying the terms encountered in both documents. The document that contains those terms are mentioned as posting lists and the frequency of those terms in the collection are called as term-frequency. When given a search query the term is looked in the index and all the documents that are contained in the index are given as result.

3. Query Module

This component process the user query by converting the user entered query into a set of keyword. This module uses the index to find the pages that contains those keywords specified in the query which are then given as input to the ranking module.

4. Ranking Module

This is an important component in any information retrieval system as this component returns the ranked set of result pages according to the relevance of the query. This enables the user to access the required information without navigating through the hundreds of pages given as result set.

Different algorithms for ranking the web pages are as follows:

1. Page-Rank Algorithm[30]

The Page Rank algorithm also known as random-surfer model was proposed by Sergey Brin and Larry Page. Page rank is based on linked structure of the web and, in this approach, a web page score is ranked high if it pointed by other high rank web pages. Page rank [42] is defined as in equation 2.1.

$$PR(A) = (1-d) + d(PR(T_1)/c(T_1) + ... + PR(T_n)/C(T_n)) \quad (2.1)$$

Where PR (A) = page Rank of A

PR (T_i) =Page ranks of pages T_i which links to page A

C (T_i) = number of outgoing links from page Ti

D=damping factor

The damping factor is set to 0.85. The page rank calculation is an iterative computation and it is calculated as a sum of page ranks of all inbound link pages linking to it dividing by the number of outbound links in each of those pages.

2. HITS (Hypertext Induced Topic Search) [43,44]

This algorithm was introduced by Jon Kleinberg. There are two types of pages-

(i). Authority page- A web page is considered as authority if it has many important pages linking to it.

(ii). Hub page- A web page is considered as hub if it contain link to many important pages.

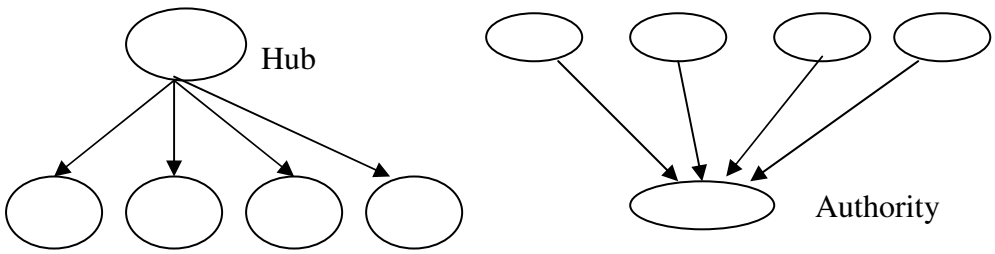

Figure 2.4 Hub and Authority nodes

Figure 2.4 shows the hub node, which is linked to many important nodes and authority referring to node which has many important web pages linked to it. The hub and authority score are updated for each iteration. The previous iteration of hub score

is used to update authority score and current iteration of authority score is used to calculate the current hub score. The parameters for authority and score are given in equation 2.2 and equation 2.3

$$x^p = \sum_{(q,p) \in E} y^q \quad (2.2)$$

$$y^p = \sum_{p,q \in E} x^q \quad (2.3)$$

Where authority score of a page p is x^p and hub score is y^p.

2.1.3 Problem with Current Web Information Retrieval Model

Traditional Web search engines provides the results by keyword based matching i.e. when a user fires a query, the query terms are searched in the index and the results are returned to the user. Due to this the web search engines sometimes provides the ambiguous results. For example

(i). The search for term "jaguar" results in the web pages related to vehicle jaguar as well as animal.

(ii). The search for the term "python" produces more results related to animal python whereas user may be interested in python programming

(iii). The search for the term "car" does not include the web pages which has automobile mentioned in it.

Matching the keywords results in ambiguous results due to the following identified reasons:

(i). This is the problem because the underlying language used for representation of web contents in traditional model is HTML which is just a presentation language and does not have any significance of the tags associated with it, which are just used as presentation tags.

(ii). The problem with the current WWW is that the contents available on the web are not machine-processable.

To process the contents of web, natural language understanding (NLP) technique which processes the contents of the web for finding semantics happens to be a very difficult technique.

Therefore, there arises the need for the web which has a meaningful contents associated to it that can be machine-processed. Tim-Berner Lee introduced the Semantic Web [45] as a technology that makes the web more intelligent by making it machine-processable.

2.1.4 Introduction to Semantic Web

"The Semantic Web is an extension of the current web in which information is given well defined meaning, better enabling computers and people to work in cooperation [46]".

This vision of the Semantic Web was proposed by Tim-Berner Lee, stating that contents on the web will be more structured letting the machine to understand the semantics of the contents. This will make the machine to understand, interpret and present more precise results, making the machine more intelligent. The Semantic Web enables intelligent services for machine processable web such as information brokers, search agents and information filters which offers greater functionality and interoperability than current standalone services [47].

2.1.5 Difference between Traditional Web and Semantic Web [48]

The difference between both the web based on various parameters has been discussed in Table 2.3.

Table 2.3 Difference between Traditional Web and Semantic Web

Parameter	**Traditional Web**	**Semantic Web**
Language Representation	The languages used for representing the contents of traditional web are HTML, XML.	The languages used for representing the contents of Semantic Web are RDF, OWL, DAML+OIL.
Language Expressiveness	The underlying language has no expressive power; it is just used as a presentation language on the web.	The underlying language for Semantic Web has expressive constructs where concepts are represented as classes and has relationship between them.
Hyperlinking	The hyperlinking structure of traditional web is defined with < link href= " ">which does not have necessarily a relevant linking.	The hyperlinking structure of Semantic Web is defined with constructs such as <rdf:seeAlso>,<owl:Imports>.

Concept	Traditional web is based on keyword-based matching of the query with a document.	Semantic Web has a concept-Based matching of the query with a document.
Machine Accessible	Traditional Web does not support machine accessibility of contents,	Semantic Web supports machine accessibility of contents.
Rule Languages	Traditional Web do not support any rule language for accessing the contents.	Semantic Web contents can be accessed via various rule languages. For example SWRL.
Ranking	Traditional Web supports Page Rank algorithm.	Page Rank algorithm as such cannot be implemented for Semantic Web because of the underlying knowledge representation.

The above Table 2.3 depicts the difference between the Traditional Web and the Semantic Web based on various parameters.

Semantic Web is the next generation web in which the information is represented with well-structured languages like RDF, OWL, supporting the concepts to be linked to other concepts allowing the machine to understand the underlying semantics of the represented information.

The next section discusses about the layered architecture of Semantic Web in detail.

2.2. SEMANTIC WEB ARCHITECTURE

World Wide Web Consortium (W3C) organization is leader in developing standards and technologies for Semantic Web. Tim-Berner Lee, Director of W3C proposed layered architecture of Semantic Web as shown in Figure 2.5. In the Semantic Web, layered architecture each layer follows the below mentioned principles-

1. Each layer in the architecture can use the technologies or support of the layers below it.
2. The upper layer is compatible with layers below it. For example, the semantics of the OWL should be fully compatible with the RDF.
3. The level of semantics increases from the lower layers as compared to the upper layers. For example OWL has more expressive semantics than RDF.

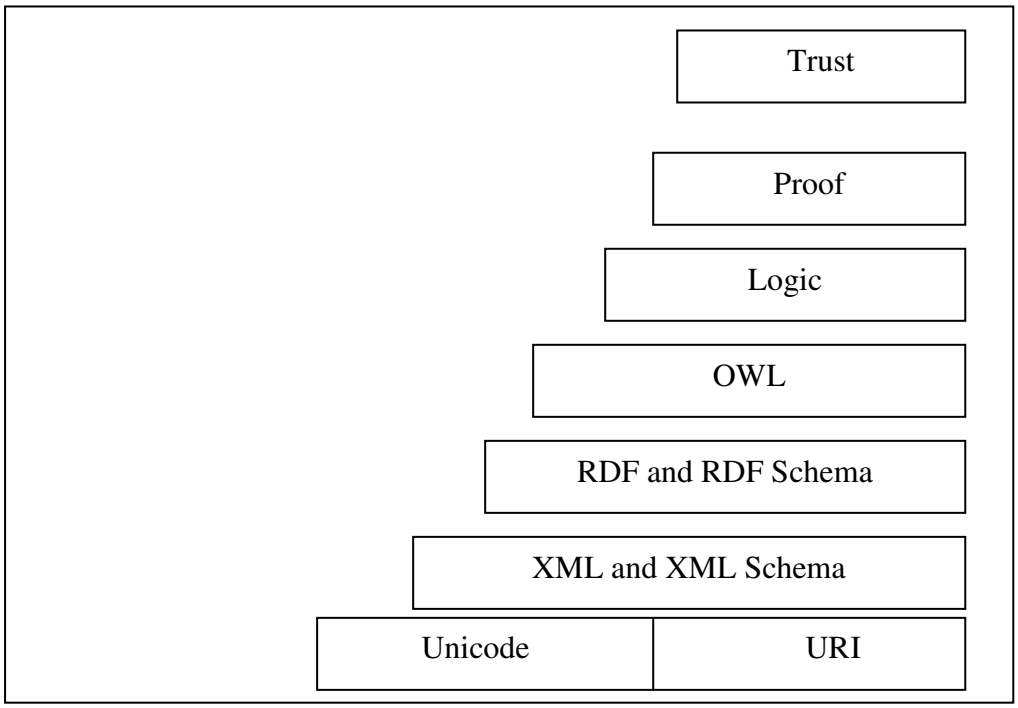

Figure 2.5 Semantic Web Layered Architecture

The foundation layer of architecture consists of Unicode and URIs which identifies the resources. Above this layer, three layers represent language representation for the specifying Semantic Web documents. These three layers from bottom to top, has more expressiveness and inferential capability constructs in comparison to the layer below it. The layers above the language layers is Logic Layer which is responsible for adding logic to the system which is the most promising capability of this Semantic Web which differentiates it from other systems. Proof layer is responsible for justification for the results produced. Trust is applicable through various technologies like digital signatures [49].

The detailed descriptions of the following components are described as following:

1. Unicode and URIs

This layer is used to identify objects by unique identifiers in the Semantic Web and to ensure that the technologies are applicable to all languages. URI (Uniform Resource Identifier) is the standard used for identifying and locating resources on the web.

2. XML [50]

XML is a popular language for exchanging information which conforms to a well-defined syntax. In XML language user can define its own tags for the structuring of the document. XML is well differentiated from HTML language as

(i). XML does not have fixed well defined tags as HTML. In XML user can define their own tags

(ii). HTML is a language primarily used for data presentation and formatting, whereas the goal of XML is to well describe the data content.

A XML document consists of text mark-up with user-defined tags for representation. Consider an example of XML document

```
<?xml version= "1.0" encoding= "UTF-8"?>
<website description>
        <name> "University of Technology" </name>
        <director>Dr. Prinayar</ director>
        <address> Delhi</address >
        <academiccourses>Mtech, Phd</academiccourses>
        <researchareas> Aeronautics, mechanical</researchareas>
        <contactno>011-23456 </contactno>
</website description>
```

Figure 2.6 Example of XML describing a web page

Figure 2.6 specifies an XML code in which the first statement describes the version and encoding scheme of the document called as *prolog* of the XML document. This document starts with user defined tag and has elements *name*, *director*, *address*, *academiccourses, researchareas*, *contactno* associated with it.

3. RDF (Resource Description Framework) [51]

RDF is a framework for processing metadata so that it can be exchanged between applications without loss of meaning. It is a method for representing information about resources on the web [52]. The goal of RDF is to enable applications to exchange data on the web while still preserving their original meaning as opposed to

HTML and XML. The main intention is not only to display documents correctly but also to allow for further processing and recombination of information contained in them. [53].

4. RDF Schema [54]

RDF schema can be viewed as a primitive language for organizing web objects into hierarchies. Key primitives are classes and properties, subclass and subproperty relationships and domain and range restrictions. RDF Schema is based on RDF [10].

5. Logic and Proof Layer [55]

Logic layer is responsible for providing logical reasoning. In this layer, logic is added to the semantic contents. Logic adds the reasoning capability to the underlying language, which is considered as the most powerful capability for any system. Adding the logic to the contents helps the software agents to process the information. With the added facts and rules, new facts can be derived from the existing knowledge. Consider a knowledgebase

Facts
1. John is father of Mary
2. Mita is mother of Mary
Rule
1. father(x, y)^mother(z,y)->husbandOf(x,z)
2. mother(x)->female(x)
With the knowledge base and the rules defined it can be derived
1. *John is husbandOf Mita*
2. *Mita is Female*

Figure 2.7 Example of Logic

Figure 2.7 specifies an example of logic which specifies the **Facts** and **Rule** and the **Facts derived** from the given knowledge. Logic added to the knowledge base makes the language more expressive and makes machine-processable.

The Proof Layer involves the actual deductive process as well as representation of proofs in web languages (from lower levels) and proof validation [49].

6. Trust layer

This is the topmost layer which is responsible for providing trust on data, contents, applications deployed in the lower layers. Trust layer emerge through the use of digital signature and other kinds of knowledge based on recommendation by trusted agents on rating and certification agencies and consumer bodies [49, 56].

The next section discusses about the technologies of Semantic Web in detail.

2.3 TECHNOLOGIES FOR SEMANTIC WEB

The technologies and tools that are recommended by W3C to make the Semantic Web a reality are metadata, ontology development tools and ontology languages. The details of these technologies are as follows:

2.3.1 Metadata [49, 57]

Tim Berners-Lee's vision for WWW is termed the "Semantic Web," where semantic metadata are the building blocks. By annotating or enhancing documents with semantic metadata, software programs (agents) can automatically understand the full context and meaning of each document and can make correct decisions about the exact user of documents and the way in which these documents should be used.

Metadata generation tools generate metadata from web resources and store it in RDF for later use. There are various metadata tools to create metadata for already existing web pages. Such tools for creating metadata are

1. Reggie [58, 59]

It is a tool capable of extracting metadata from given web pages. The user can select any existing schema file or can create his/her own schema files. Reggie extracts the META tags from a given URL and attempts to add them to the most appropriate fields of the chosen schema. It also allows users to create their own metadata schema files.

2. DC-DOT [58, 60]

It is another tool for metadata extraction. In comparison to Reggie where the user provides the schema file, DC-DOT specifically uses the Dublin Core schema, a metadata element set for description of electronic resources, to extract metadata from

a given web resource. DC-DOT uses the information contained in the META tags of a Web source to generate the RDF model.

2.3.2 Ontology Languages

There have been a lot of research efforts in the representation language for Semantic Web. Semantic Web documents can be represented with XML/XML Schema but the tags are not machine processable. Afterwards RDF /RDFS were used to represent documents, which have certain limitations which are overcome by using OWL. RDF language representation does not have expressive constructs as compared to OWL.

2.3.3 Ontology Development tools

There are different Ontology development tools for the development of ontology. Some of the ontology development tools are commercial whereas some of them are open source. Most of the tools are research efforts and are available as open source. For example, the open source tools are Protégé [61, 62], NeOn Toolkit [63, 64], Apollo [65, 66], WebODE [67], OilEd [176], OntoEdit [177] and commercial available tool is TopBraid Composer [171].

2.4. ONTOLOGY

2.4.1 Definitions

The ontology has been defined by different researchers as -

(i). "An artefact, constituted by a specific vocabulary used to describe a certain reality plus a set of explicit assumptions regarding the intended meaning of vocabulary" [68, 69].

This was the definition proposed by Guarino[69] in 1998 before ontology was applied in the domain of knowledge representation and knowledge retrieval. This specifies that ontology is a vocabulary set that specifies a real world scenario and has certain assumptions that are made to form that vocabulary.

(ii). "Ontology is an explicit specification of conceptualization" [69, 70, 75].

This definition was given by Gruber [70] in 1993. 'Explicit' means the concepts, properties, functions, axioms must be clearly defined, and 'conceptualization' means abstract model of some phenomenon/application in the real word that can be well represented with concepts and the relation between the different concepts.

(iii). "Ontology is formal specification of a shared conceptualization" [69, 71, 75].

This definition was given by Borst[71] in 1997. 'Formal' specifies ontology should be machine readable. 'Shared' means consensual knowledge which means that ontology should be such designed that it could be shared with different application.

(iv). "Ontology is a formal, explicit specification of a shared conceptualization" [72, 75].

This was the definition given by Studer [72] in 1998, which was based on the merging of the earlier definitions given by Gruber [70] and Borst [71].

(v). "Study of categories of things that exist or may exist in some domain. The product of such study called ontology is catalogue of types of things that are assumed to exist in a domain of interest D from the perspective of a person who uses a language L for the purpose of talking about D"[73].

This definition proposed by Sowa [74] in 2004 refers to that for defining domain it can be categorized into classification. Hence ontology is a catalogue of things for a domain which can be defined in a language.

2.4.2 Reasons for developing Ontology

It is the most important component of the Semantic Web which is used to represent domain knowledge. It describes a set of concepts and relationship between those concepts in a specified domain. The following reasons have been identified for developing ontology [75]:

(i). To share common understanding of the structure of information among people or software agents.

Ontology enables the concepts to be defined in a way that can be shared by people or agents. For example, if several websites contain information about a product and these

websites share the same ontology then agent must be able to aggregate the information about the product from the different sites and present to the user or any required application.

(ii). To enable reuse of domain knowledge

To design ontology from scratch is a tedious and time consuming task, hence ontology defined for a domain must be designed to cover the concepts so that it can be reused /extended by some application rather than creating. These created ontology can be shared by keeping them in a ontology repository.

(iii). To make domain knowledge assumptions explicit

Explicit specification for domain knowledge makes it easy to change the assumption if the knowledge of the domain changes. It easily allows a new user to understand the domain terms easily.

(iv). To separate knowledge from the operational knowledge

It is better idea to separate operational knowledge from the domain knowledge from the knowledge management perspective because it leads to an inefficient system as, such a design hinders knowledge engineers ability to express deeper relationship among knowledge items [76].

(v). To analyse domain knowledge

Ontologies are used to explain a domain completely with concepts, properties and relations that exists between them. Such a formal specification helps in analysing a domain explicitly and allows knowledge reuse.

2.4.3 Ontology languages

There are various languages in which ontology can be specified. The language specifies the formal semantics of a language. The language adds the expressiveness to the representation of knowledge allowing the inferences and reasoning support making the semantics of the language machine-accessible. The language stack for Semantic Web [77] is as given in Figure 2.8.

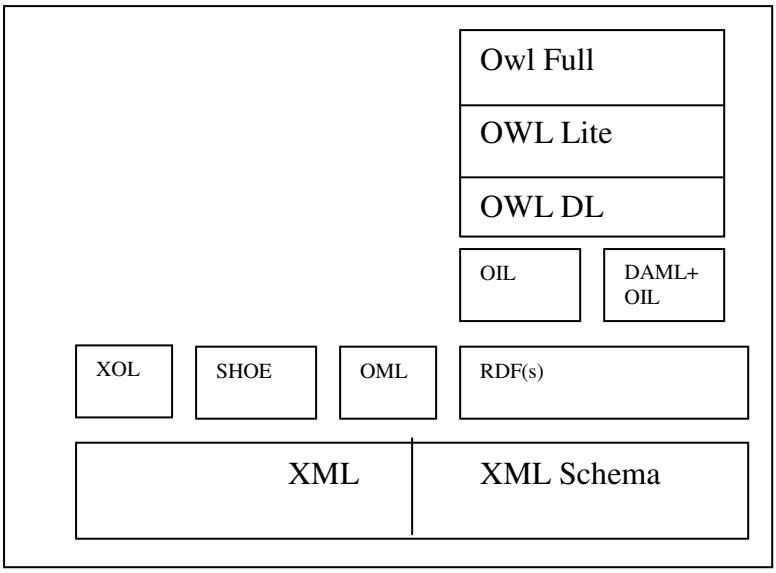

Figure 2.8 Languages Stack in Semantic Web

The various ontological modelling languages are

1. XML & XML Schema [10]

XML is a meta language which allows the user to define its own tags. XML acts as a uniform data exchange format between applications. XML Schema are used to define new schemas from other defined schemas.

2 XOL (An XML-Based Ontology Exchange Language) [78]

XOL (XML Ontology Exchange Language) is a frame based language with an XML syntax that is currently being designed for the exchange of ontologies for molecular biology [79]. Some of the predefined used in XOL language are specified in Table 2.4.

Table 2.4 Some Constructs of XOL language

<class>	Start of tag for defining class
</class>	End of tag for defining class
<slot>	Defines the range
Type-of	Instance indicating type of class
Instance-of	Indicates instance of class
Subclass-of	Defines a subclass of class
Minimum-cardinality	Specifies minimum cardinality for a slot
Maximum-cardinality	Specifies maximum cardinality for a slot

Figure 2.9 shows an example of a class university having slot value for academic staff not to be more than 100.

```
<class>
<organization>University<//organization>
<slot>
<name>MM University</name>
<staff>academic staff<staff>
<value-type>integer</value-type>
<numeric-max>100</numeric-max>
</slot>
</class>
```

Figure 2.9 An example of XOL Language

Figure 2.9 specifies a document in XOL language for a *University* specified with <*class*> tag and the range is specified using <*slot*> tag which has user defined tags within it *(*<*organization*>*,* <*name*>*,* <*staff*>*,*<*value-type*>*,*<*numeric-max*>*).*

3. SHOE (Simple HTML Ontology Extension) [80]

SHOE was developed at the University of Maryland. This was the first ontology description language created for the Semantic Web called as Simple HTML Ontology extension. It has basically syntax of HTML which is extended with new tags to semantically annotate the web pages. Consider an example of an ontology defined with SHOE tags defined in [81, 82, 83]. This is a well-defined language for knowledge annotation of web pages. It has knowledge annotator graphical user interface tool for embedding SHOE annotations which is used for automatic annotation of the pages [84].

The process of knowledge annotation with SHOE constructs has three phases –

1. Development of a Ontology with SHOE syntax.
2. Annotating the HTML web pages with the developed ontology.
3. Allow the agents to retrieve the information.

```
<HTML>
<HEAD>
<META HTTP-EQUIV= "SHOE" CONTENT = "VERSION=1.0">
<TITLE> CS Department Ontology </TITLE>
</HEAD>
<BODY>
<! Declaring ontology name and version>
<Ontology id= "CS-Dept-Ontology" version= "1.0"
<! Declaration of ontology being borrowed from another ontology>
<USE_Ontology id= "base-ontology" version= "1.0" Prefix= "base"
URL=http://www.cs.umd.edy/projects/plus/SHOE/base.html>
<! Defining categories>
<DEF-CATEGORY NAME= "Department" >
<DEF-CATEGORY NAME= "Worker" ISA= "Person">
<DEF-CATEGORY NAME= "Faculty" ISA= "Lecturer">
<! Relationships between categories>
<DEF-RELATION_NAME= "Advisor">
<DEF-ARG POS= "1" TYPE = "Student">
<DEF-ARG POS="2" TYPE= "Person">
</DEF-RELATION>
</ONTOLOGY>
```

Figure 2.10 Example of SHOE Ontology for CS Department

This example specifies the ontology of *CS-Dept-Ontology* defined using the SHOE language constructs. The different tags of SHOE syntax in the above example specifies (refer Table 2.5):

Table 2.5 Different tags of SHOE

<Ontology id= *"CS-Dept-Ontology" version= "1.0"* *</ONTOLOGY>*	Ontology appears within these tags which specifies name of ontology, id and version.

<DEF-CATEGORY NAME>	This tag make category definition that specify the categories under which various instances could be classified
<DEF-RELATION>	This is used to make relational definitions that specify format of n-ary relational claims that may be made by instances regarding instances and other data.

Table 2.5 defines different tags specified in Figure 2.10 which defines CS-department ontology that defines categories using *<DEF-CATEGORY> (Department, Worker. Faculty).* Relationships exist between different instances using *<DEF-RELATION> (Advisor* relation exists between *Student* and *Person*).

4. OML (Ontology Markup Language) [77]

OML language was developed at University of Washington as XML serialization to the SHOE language. Therefore, OML and SHOE language share many features. OML is explicitly oriented towards the abstract semantics. All the features of XML Schema are also in OML. Consider an example of OML adapted from [79]:

"Ralph Swick says that Ora Lassila is the creator of the resource *http://www.w3.org/home/Lassila*".

```
<OML>
<Type.Object name = "Document"/>
<Type.Object name = "Person"/>
<Type.BinaryRelation name = "create"
source.Type = "Person" target.Type = "Document"/>
<Type.BinaryRelation name = "attributedTo"
source.Type = "Proposition" target.Type = "Person"/>
<Type.Relation name = "Create" binrel = "create">
/* reified relation */
<Type.Function name = "agent" target.Type = "Person"/>
<Type.Function name = "theme" target.Type = "Document"/>
</Type.Relation>
/* instances in a collection */
<Document id = "ora-homepage" about = "http://www.w3.org/Home/Lassila"/>
<Person id = "ora" text = "Ora Lassila"/>
<Person id = "ralph" text = "Ralph Swick"/>
<Instance.ReifiedBinaryRelation id = "ora-create-homepage"
source.Instance = "ora-home-page"    /* subject */
target.Instance = "ora"/>                /* object */
<classification type = "creator"/>    /* predicate */
<attributedTo target.Instance = "ralph"/>
</Instance.ReifiedBinaryRelation>
```

Figure 2.11 Example of OML

Figure 2.11 shows an example of OML in which different types (objects and relations) in ontology are defined using <*Type. Object name*>, <*Type. Binary Relation name*>, <*Type. Relation name*>. Reified [10] relations are defined using <*Type. function name*>. Instances are defined for each object name. For each reified relation Subject, Predicate, Object are defined.

5. RDF [85]

It is a foundation for processing metadata; it provides interoperability between applications that exchange machine-understandable information on the web .The RDF language representation has mainly three main components [10, 86]-

(i).Resource

It is anything that can have a URI; this includes all the web pages as well as individual elements of a document. A URI is a character string that identifies an abstract or physical resource on the web. Examples of different URI schemes are as [87]:

(i). A URI following the FTP scheme for file transfer protocol services

ftp://ftp.mysite.com/files/foobar.txt

(ii). A URI following the HTTP Scheme for Hypertext Transfer protocol services

http://www.mysite.com/pub/foobar.html

(iii). A URI following the MAILTO scheme for email addresses

mailto:em@w3.org

(ii). Property

It is a resource that has a name and can be used as a property, for example author or title.

(iii). Statement

It is the combination of a resource, property, and a value. These combinations are also known as 'Subject',' Predicate' and 'Object' of a statement and are represented as <S, P, O>. Statements can be represented as graph. For example Figure 2.12 shows the statement ***Vandana is PhD Student of YMCA University of Science & Technology*** in <S,P,O> format.

Subject	*Vandana*
Predicate (property)	*#PhDstudent*
Object (literal)	*http://ymcaust.ac.in*

Figure 2.12 Dividing the RDF statement into <S, P, O>

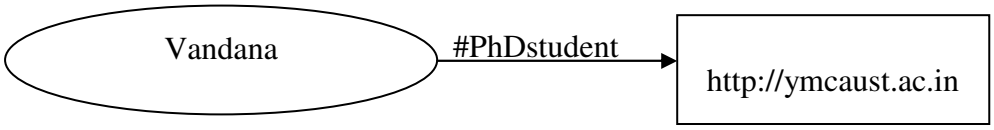

Figure 2.13 RDF Graph Representation of statement

Figure 2.13 represent the graph representation of the statement ***Vandana is PhD Student of YMCA University of Science & Technology*** where Subject (*Vandana*) is connected to Object*(http://ymcaust.ac.in)* through Predicate(#PhDstudent).

6. OIL (Ontology Interchange layer/Ontology Inference layer) [88, 77]

OIL is a web ontology language based on RDFS which supports a well-defined semantic vocabulary and reasoning constructs for ontology development. OIL was developed as a research product of European Union Project (EU). Consider a simple example of ontology defined in OIL language specification where different classes are defined using OIL language constructs

```
class-def mammals
class-def humans subclass-of-mammals
class-def animal subclass-of mammals subclass-of NOT human
class-def plant subclass-of NOT human  subclass-of NOT animal
class-def defined herbivore
        subclass-of-animal
        NOT herbivore
        slot-constraint eats
        value-type plant
```

Figure 2.14 Example of OIL defining classes

Figure 2.14 shows example of OIL defining classes where different classes are defined using *class-def* constructs and subclass using *subclass-of* construct(*humans* is subclass-of *mammals*) and property(*eats*) constraints, its value(*plant*) using *slot-constraint, value-type.*

OIL is based on following three elements for representation [89]:

(i). **Frame Based Systems**

These form the modelling primitives for OIL language. It supports the primitives with which classes, superclass, relations and properties (slots) can easily be described to specify a ontology using OIL.

(ii). **Description Logics(DL)**

The knowledge can be represented in form of concepts and roles (slots) which enables reasoning support for the language. An important aspect of DL is that it has well understood theoretical properties and meaning of any expression of DL that can be described in mathematical precise way which enables reasoning with concept descriptions and automatic derivations of classification taxonomies [90].

(iii). **Web Standards**

OIL is based on the web standards of W3C that has syntax of XML, RDF and RDFS hence; it has the compatibility with the existing web standards to enable its use on the web.

7. DAML (Darpa Agent Markup Language) [91]

The DARPA (Defence Advanced Research project Agency) Agent Markup Language was started in 2000. The initial version of DAML was called as DAML-ONT but with the research effort of EU/US, it emerged into DAML+OIL, and is considered as more robust language than RDF and RDFs. The language has the constructs for knowledge representation. DAML language specification is built upon RDF and supports Description Logics which has the inference constructs. Certain language constructs of DAML are defined in Table 2.6

Table 2.6 Language Constructs of DAML

Construct	**Definition**
Daml:restriction with daml:onProperty	Specifies a slot being restricted on the property specified
Daml:intersectionOf	Disjunction of class expression
Daml:unionOf	Conjuction of class expression
Daml:complementOf	Negation of Class expression
Daml:mincardinality	Minimum cardinality constraint on a Property
Daml:maxcardinality	Maximum cardinality constraint on a Property
Daml:transitiveProperty	Specifying the transitive property
Damlinverseof	Specifying the inverse property

Example of DAML

```
<daml:Class rdf:ID="child">
        <daml:subClassof rdf:resource = "#person" >
                <daml:restriction>
                        <daml:onProperty rdf:resource= "#hasMother"/>
                        <daml:cardinality>1</daml:cardinality>
                </daml:restriction>
        </daml:subClassof>
        <daml:subClassof>
                <daml:restriction maxcardinality="2">
                        <daml:onProperty rdf:resource= "#hasParent"/>
                        <daml:cardinality>2</daml:cardinality>
                        <daml:Class>
                        <daml:unionOf rdf:parseType="daml.collection">
                        <daml:Class rdf:about = "Father"/>
                        < daml:Class rdf:about = "Mother"/>
                        </daml:Class>
                </daml:restriction>
        </daml:subClassof>
</daml:Class>
```

Figure 2.15 Example of DAML

This example in Figure 2.15 describes ontology in DAML for a class *child* which is subclass of class *person*. It specifies that the child can have one mother by limiting the cardinality on property *#hasmother* to one. Property *#hasparent* has cardinality two, which is specified by <*daml:unionOf*> construct, specified with class *mother* and *father.*

8. OWL [92]

OWL Language has emerged from DAML+OIL language on the recommendation of W3C. This layer adds more vocabulary than RDF for describing properties and classes; relations between classes (e.g. disjointness), cardinality (e.g. exactly one), equality, richer typing of properties, characteristics of properties (e.g. symmetry) and enumerated classes [8]. OWL provides three increasingly expressive sublanguages: OWL Lite, OWL DL, OWL Full [94]. The features of these three sublanguages are as follows [93]:

(i). **OWL Lite [93, 10]** supports those users primarily needing a classification hierarchy and simple constraints. For example, while it supports cardinality

constraints, it only permits cardinality values of 0 or 1. It should be simpler to provide tool support for OWL Lite than its more expressive relatives, and OWL Lite provides a quick migration path for thesauri and other taxonomies. OWL Lite also has a lower formal complexity than OWL DL.

(ii). **OWL DL [93, 10]** 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 so named due to its correspondence with Description Logics, a field of research that has studied the logics that form the formal foundation of OWL.

(iii). **OWL Full [93, 10]** is for users who want maximum expressiveness and the syntactic freedom of RDF with no computational guarantees. For example, 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.

```
<rdf:RDF
        xmlns:rdf=http://www.w3.org/1999/02/22-rdf-syntax-ns#
        xmlns:rdf=http://www.w3.org/2000/01/rdf-schema#
        xmlns:owl=http://www.w3.org/2002/07/owl#>
<owl:Ontology rdf:about=”xml:base”/>

<owl: Class rdf:ID= “Professor”>
        <rdfs:subClassof rdf:resource= “Person”/>
<owl: class>
<owl:Class rdf:ID= “Student”>
        <owl:disjointWith rdf:resource= “Professor”/>
</owl:Class>

<owl:Class rdf:ID= “faculty”
        <owl:equivalentClass rdf:resource= "#Profesor”/>
</owl:class>
<owl:Class rdf:about= “Course”>
        <rdfs:subClassof>
                <owl:Restriction>
                        <owl:onproperty rdf:resource “#istaughtby”/>
                        <owl:hasvalue  rdf:resource = “#faculty”/>
                </owl:Restriction>
        </rdfs:subClassof>
</owl:Class>
```

Figure 2.16 University ontology (some concepts) specified in OWL language

University Ontology (refer to Figure 2.16) specifies following classes:

1. *Professor*
2. *Student*
3. *Faculty* equivalent to class *Professor*
4. *Course,* where course having restriction that it can be taught by faculty

Table 2.7 Comparison of different Ontology Languages

Ontology Language	**Language Specification**	**Developed at**	**Basic concept**
XOL[78]	Ontology exchange Language	US Bioinformatics Community	Extensions of XML constructs
SHOE[80,81,82,83,84]	Simple HTML Ontology extension	University of Maryland	Extension of HTML semantics with the semantics added to the constructs
OML[79]	Ontology Markup Language	University of Washington	Extension of XML Schema
RDF[85,86]	Resource Description Framework	W3C	Extension Of XML constructs
OIL[88,89,90]	Ontology Interchange Language	W3C	Syntax and semantics based on XOL,RDF
DAML[91]	Darpa Agent Markup language	Joint committee from US and European Union	Extension of RDF and RDF Schema
OWL[92]	Web ontology language	W3C	Syntax based on RDF,XML

Table 2.7 gives a comparative analysis of different ontology languages on various parameters. RDF, OWL are the most widely used ontology languages. OWL is considered to be more expressive language as compared to RDF but is more complex.

2.5 ONTOLOGY DEVELOPMENT TOOLS

Ontology development tools are used for developing ontology. There are lot of tools available for ontology development, some of them are research efforts of universities

and are freely available whereas some of them are commercially available. The detailed descriptions of some of the tools are as follows:

2.5.1 Protégé [75, 95]

Protégé is an ontology editor developed by Stanford Medical Informatics (SMI) group at Stanford University. It is an open-source platform for ontology development. It has been used for many years for knowledge acquisition of domain knowledge and for building domain ontology in recent years. The architecture of Protégé [96] is depicted in Figure 2.17

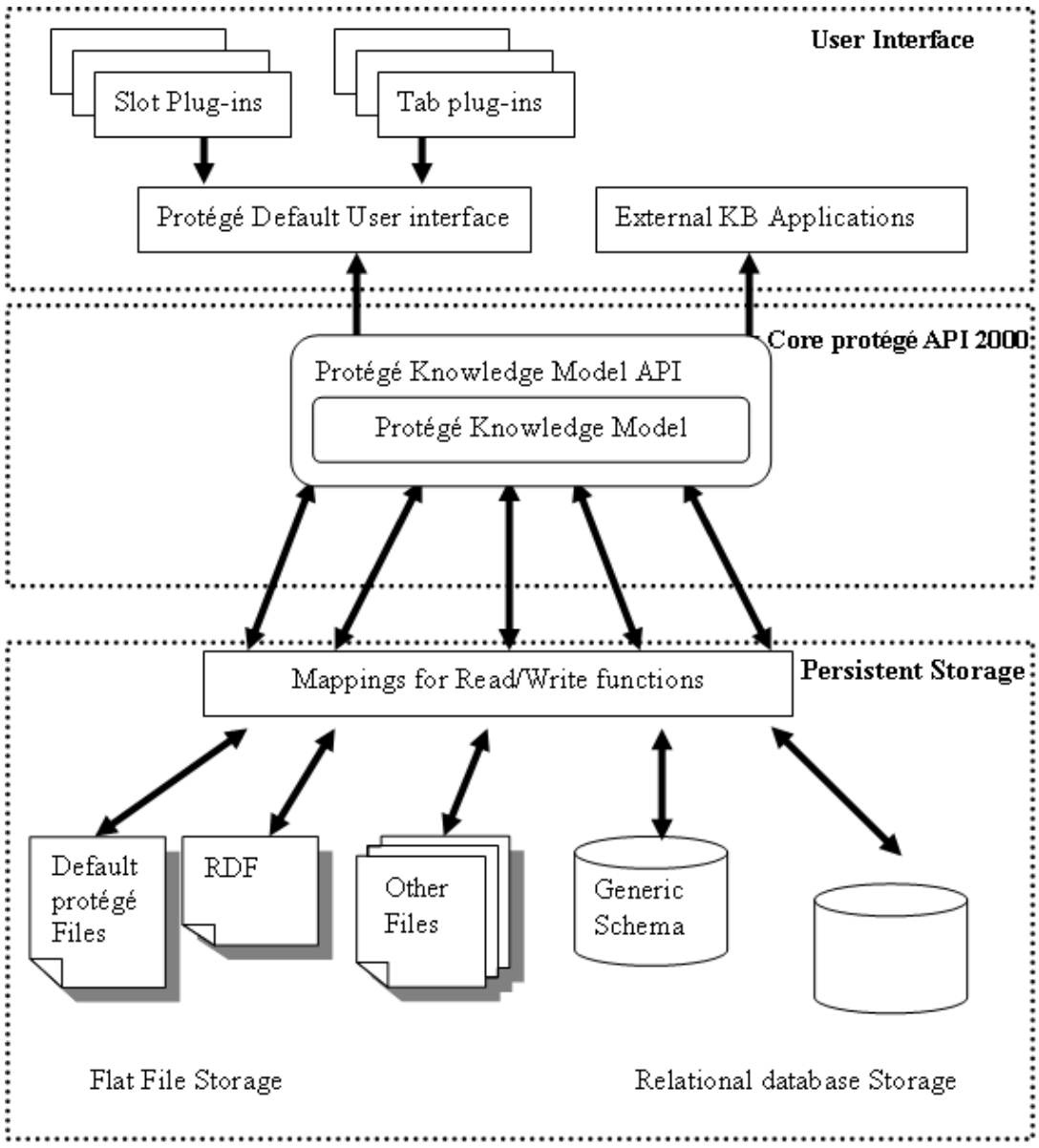

Figure 2.17 Protégé Architecture

The architecture of Protégé can be broadly divided into three parts as shown in Figure 2.17.

1. Core Protégé API 2000

The core of Protégé 2000 consists of Protégé Knowledge Model. The Protégé Model consists of knowledge base such as instances, classes etc., any access to these elements is through Protégé application interface (Protégé APIs).

2. Persistent Storage

This component is used to store the knowledge base, the schema in relational database storage and other Protégé files in flat file storage through the mapping for read/write function.

3. User interface

User Interface consists of Plug-ins, Tab-plug-ins in default user-interface. With the use of Plug-ins the development workload can be distributed across multiple programmers [96]. The plug-in architecture also help the user to incorporate the programmed Plug-ins into the architecture to make it more customizable, There are lot of slots and tab plugins available which are developed from time to time and are compatible with Protégé architecture.

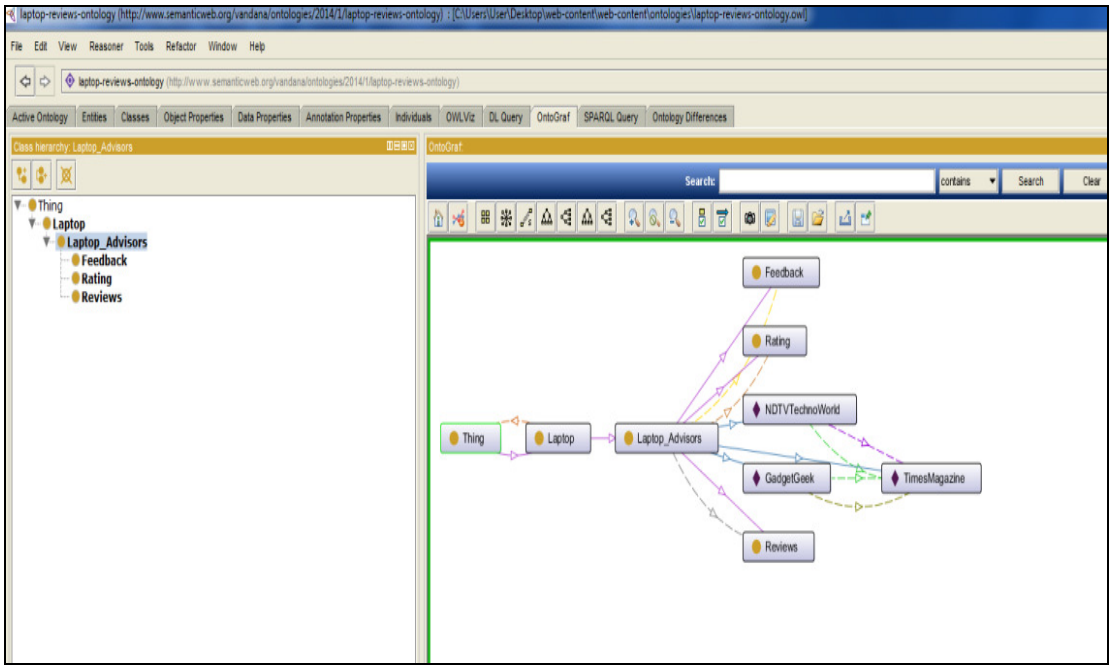

Figure 2.18 GUI Interface for Protégé

Figure 2.18 shows the default user interface of Protégé development tool used to develop ontology which has the features of creating classes, data property, object property for modelling a concept. It has the features of querying the ontology using SPARQL and DL Logic.

2.5.2 WebODE [97]

WebODE architecture is based on client server architecture. Its architecture is highly extensible and usable, as new services can be added to this architecture very easily. Ontologies are stored in WebODE in relational database. WebODE architecture supports collaborative ontology development.

The architecture of WebODE as proposed in [98] is given in Figure 2.19

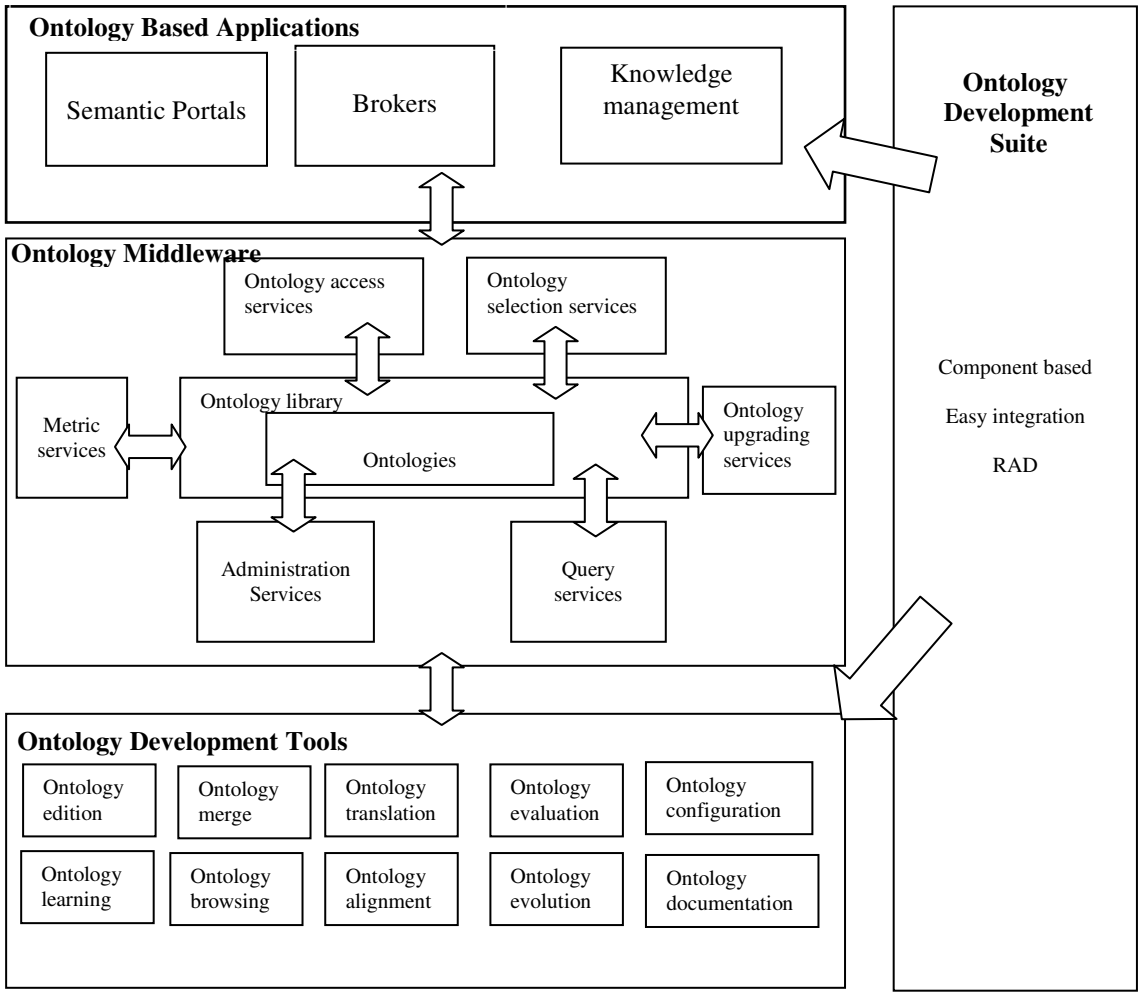

Figure 2.19 Architecture of WebODE

The main components of WebODE are discussed below:

1. **Ontology Development Tools**

   This component includes Ontology merging, Ontology translation into another language like from RDF to OWL, Ontology browsing, Ontology learning, Ontology documentation and many other ontology development tools

2. **Ontology Middleware**

   This component has a ontology repository and an interface for various services like ontology access, ontology selection, metric service, ontology upgrading service, administration service, query service provided by WebODE architecture.

3. **Ontology Based Applications**

   This component has applications which are based on ontology provided by WebODE like Knowledge Management Application, Semantic Portals which are achieved interacting with middleware layer.

4. **Ontology Development Suite**

   This component helps in Rapid Application Development (RAD) and integration of ontology application into the required systems.

2.5.3 NeOn toolkit [102]

NeOn is ontology development tool which supports multi-platform ontology engineering environment. It is built on eclipse platform and has modular architecture. NeOn is a project involving fourteen European partners and is co-funded by European Commission's sixth framework[103].Under this research effort various plug-ins have been developed for various ontology life-cycle stages for example for visual modelling (OntoModel),Ontology reuse (Watson), Ontology learning(text2Onto), Ontology Mapping(R2O,FOAM),Ontology diagnosis and repair (RaDON)[102]. NeOn toolkit has 45 Plug-ins which covers all of ontology lifecycle aspects. The architecture of NeOn toolkit is shown in Figure 2.20.

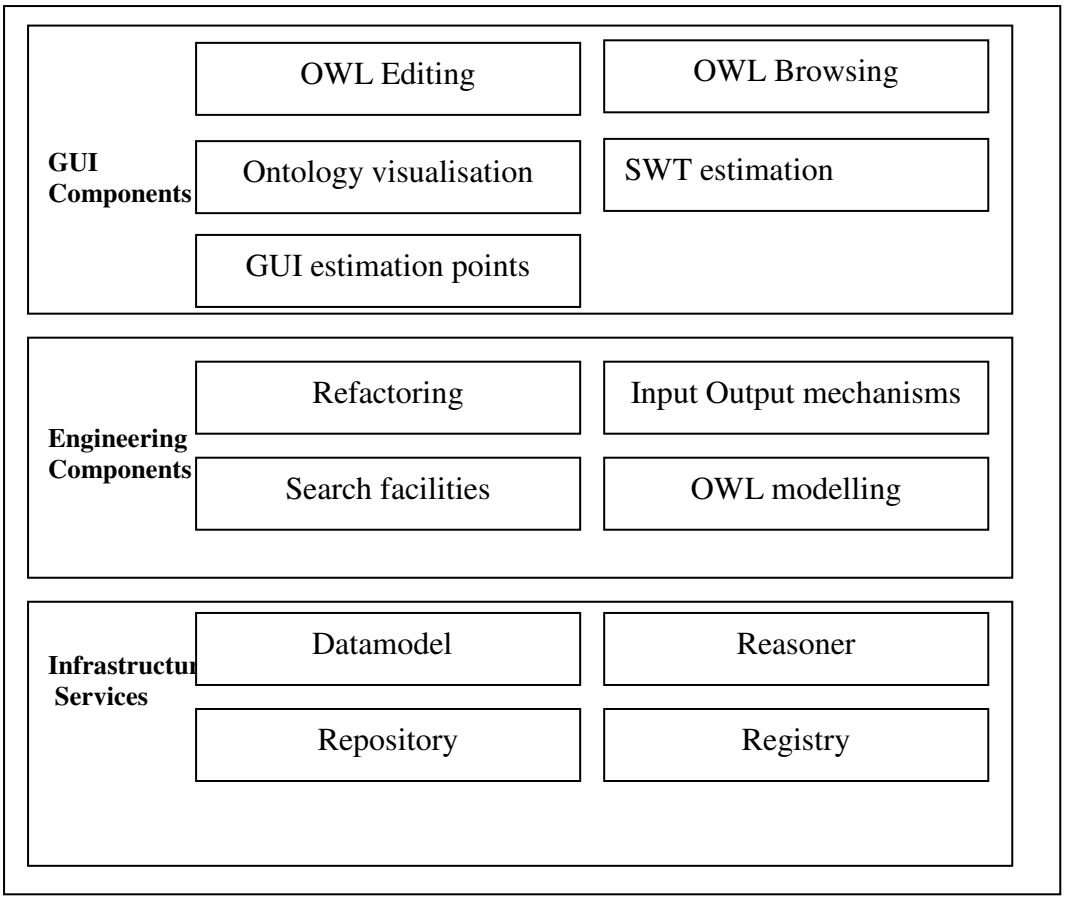

Figure 2.20 Architecture of NeOn Toolkit

The architecture of NeOn (see Figure 2.20) consists of following components:

1. GUI Components

   GUI interface of NeOn toolkit has different components for OWL editing, browsing and visualization interface component.

2. Engineering Components

   This component acts as an interface between the infrastructure services and GUI interfaces, providing different components that interact with infrastructure services for giving output to the user.

3. Infrastructure Services

   This component consists of infrastructure services provided by the architecture. For example Ontology Registry services are provided by Oyster [102,104], Ontology reasoner services are provided by KAON [105].

2.5.4 SWOOP [99]

SWOOP (Semantic Web Ontology Overview & Perusal) ontology editing tool is web based ontology editor and browser. SWOOP architecture is based on Model-View-Controller design paradigm [100]. The architecture of SWOOP is as depicted in Figure 2.21.

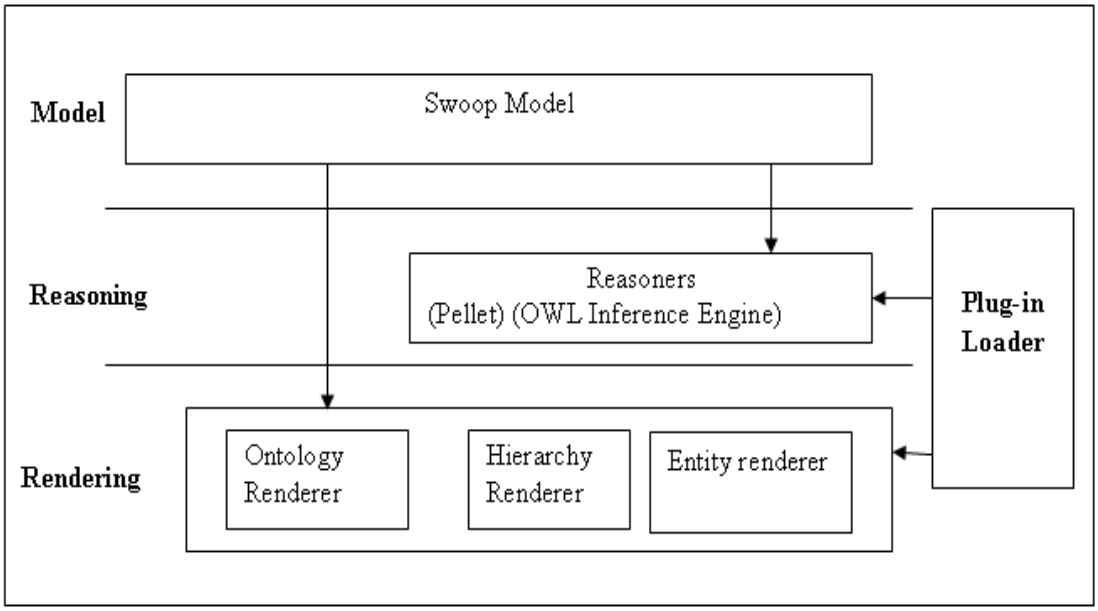

Figure 2.21 Swoop Architecture

. The architectural components of SWOOP model are as discussed below:

1. **SWOOP Model**

   This component store ontologies and ontology related information which are loaded by reasoners into the SWOOP model.

2. **Reasoners**

   SWOOP architecture has the inferential capability which is supported by integrated reasoners Pellet [101] and OWL inference engine [99,100].

3. **Rendering**

   This component is used for visualization and loads the new reasoners and renderers dynamically when required. The different level of editing and rendering

provided by SWOOP architecture are ontology rendering, hierarchy rendering and entity rendering.

4. Plug-in Based system

All the control of this architecture is handled through this component. This component is responsible for loading new reasoners or renderers dynamically into the system.

Table 2.8 Comparative Analysis of ontology editing tools

Tool	**Developed by**	**Open Source/commercial Tool**
Protégé[96]	Stanford university	Open Source
NeOn Toolkit[102]	European Research Community	Open Source
WebODE[97,98]	Technical School of Computer Science, Spain	Open Source
SWOOP[100]	MIT University	Open Source

The comparative analysis of the above discussed development tools have been shown in Table 2.8. Among these tools, Protégé is the most widely used tool for ontology development because of the plug-ins and the features it supports.

The next section discusses about the rule languages with inferential capabilities for ontologies.

2.6 ONTOLOGY RULE LANGUAGES

Ontologies are the mechanism for knowledge representation which can be specified by using different languages like RDF, RDF Schema, OWL. These languages offer a wide variety of expressiveness constructs to represent a domain. The classes, properties, property restrictions can be easily implemented using these languages. For inferential capability, that is to deduce new facts from the knowledge base various rule languages are used on these languages. The various rule languages which are widely used for inference mechanism are:

1. SWRL(Semantic Web Rule Language)[10,107]

SWRL is a rule language for the Semantic Web based on the language syntax which is a combination of OWL DL and OWL Lite with Horn logic [107]. SWRL rules are of the form antecedent-consequent pair where antecedent is referred to as body part of the rule and consequent refers to the head part of the rule. The head and body part of rule may be conjunctions of one or more atoms. A SWRL rule is of the form:

$$A_{1,}\ldots\ldots\ldots, An \rightarrow B_{1,}\ldots\ldots.., B_n$$

Where $A_1\ldots\ldots A_n$ refers to the head part of the rule and commas represent the conjunction of one or more atoms and $B_1\ldots\ldots B_n$ refers to the body part of the rule. For example consider ontology of a university knowledge base. SWRL rules for the same can be defined as specified in Figure 2.22:

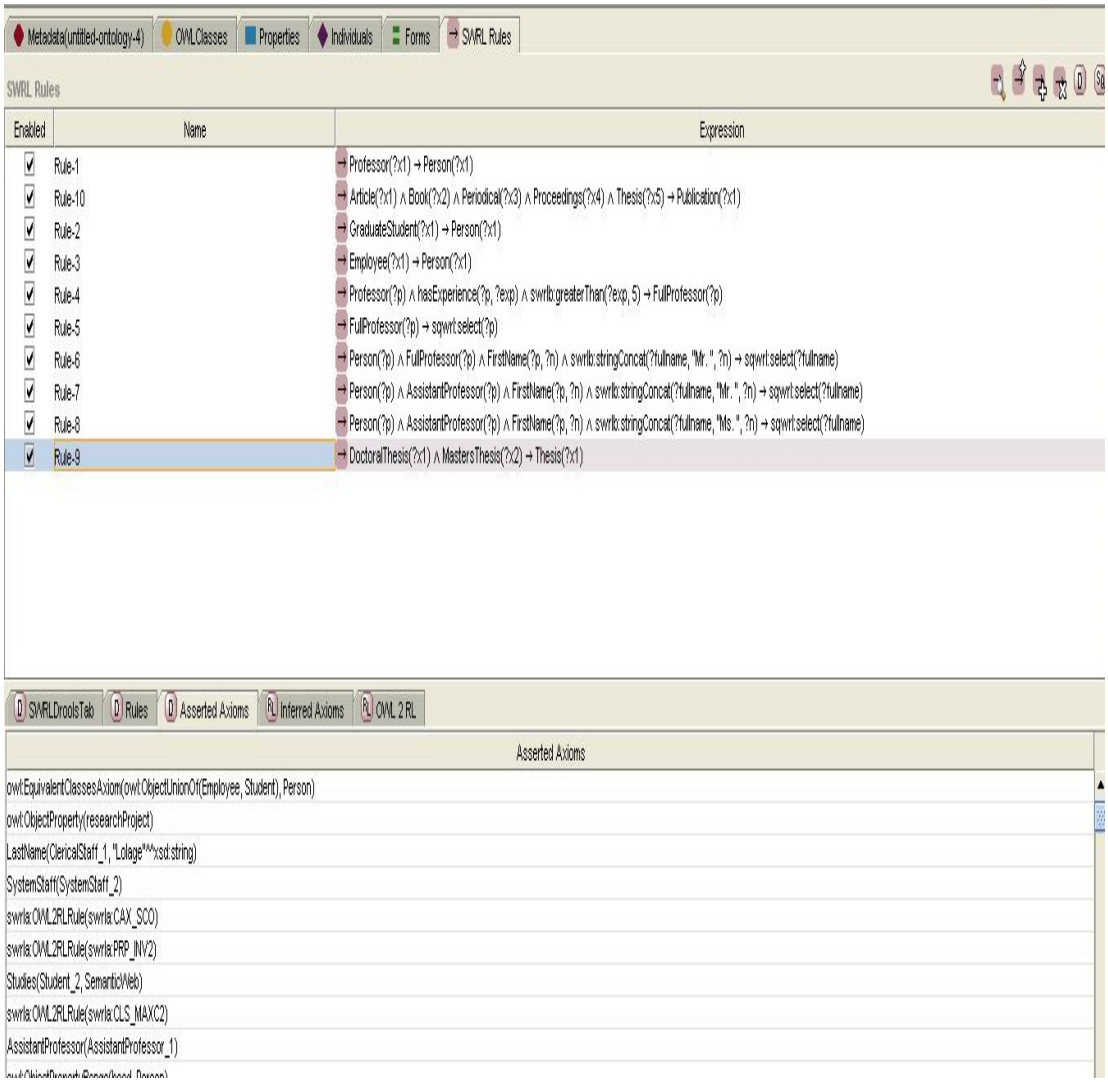

Figure 2.22 SWRL Rules implemented in Protégé

Figure 2.22 shows SWRL rules implemented in Protégé. Consider one of the rule *Professor(?p)^hasExperience(?p,?exp)^swrlbgreaterThen(?exp,5)→FulltimeProfessor* This rule specifies that a professor which has experience greater then 5 years is fulltime professor.

2. Rule Markup language (RuleML)[106]

RuleML is a Rule Markup language for Semantic Web. RuleML has four categories of rules-

(i). **General reaction rules –**

These rules are applied in the forward direction for observing/checking events/conditions and performing an action if and when all events/conditions have been perceived/ fulfilled.

(ii). **Integrity constraints rules-**

These rules are also forward-oriented, i.e. triggered by updates, mainly for efficiency reasons.

(iii). **Derivation rules**

The category of these rules can be applied in the forward direction as well as in a backward direction, the latter reducing the proof of a goal (conclusion) to proofs of all its subgoals (premises).

(iv). **Facts rules**

These classes of rules are used for an application direction.
The rule "**The customer is given 10% discount if he spends 5000Rs for a bill**" is represented in syntax of RuleML as

```
<Implies>
<head>
        <Atom>
                <Rel>discount</Rel>
                <Var>customer</Var>
                <Ind>10%</Ind>
        </Atom>
</head>
<body>
        <if>
        <Atom>
                <rel>spend</rel>
`               <Var>customer</Var>
                <ind>5000Rs</ind>
                <ind>Bill</ind>
        </Atom>
        </if>
</body>
</implies>
```

Figure 2.23 Example of Rule Language

Figure 2.23 shows an example of RuleML where it starts and ends with <implies> </implies> syntax and is divided into <head><atom> and <body> <atom>part. The relation predicate (discount, spend) is represented by <rel> tag. The variables (customer) is represented with <var> tag and constant values(10%,5000Rs) are represented by <ind> individual tag.

The next section discusses about the various semantic search engines and their architecture.

2.7 SEMANTIC SEARCH ENGINES

Semantic Search Engines searches Semantic Web data from the web which is in the form of RDF documents, OWL documents and ontologies. Semantic search engines work on highly structured languages such as RDF, OWL, DAML hence making the contents to be machine processable. The different Semantic Web search engine are described as below:

2.7.1 Ontokhoj Ontology Search Engine

Ontokhoj [108, 109] is an Ontology search and classification tool that gives a ranked list of ontologies for a given query. Ontokhoj searches and ranks Ontologies (RDF, DAML+OIL, OWL) over web. It also uses the existing Classification algorithms like K-nearest Neighbours algorithm (KNN) and Naive Bayes algorithm to classify the crawled ontologies into respective domain hierarchy which is derived from DMOZ (Directory Mozilla) Open Directory Project [110]. This tool is developed in Java having the following main functionalities:

1. **Ontology Crawling**

Ontology crawling is based on the underlying language representation which uses the concept of URIs to further crawl i.e. hyper linking concept. The crawler is designed to retrieve ontologies represented in different languages i.e. RDF, OWL, DAML+OIL format. For example Figure 2.24 shows hyper linking between RDF statements

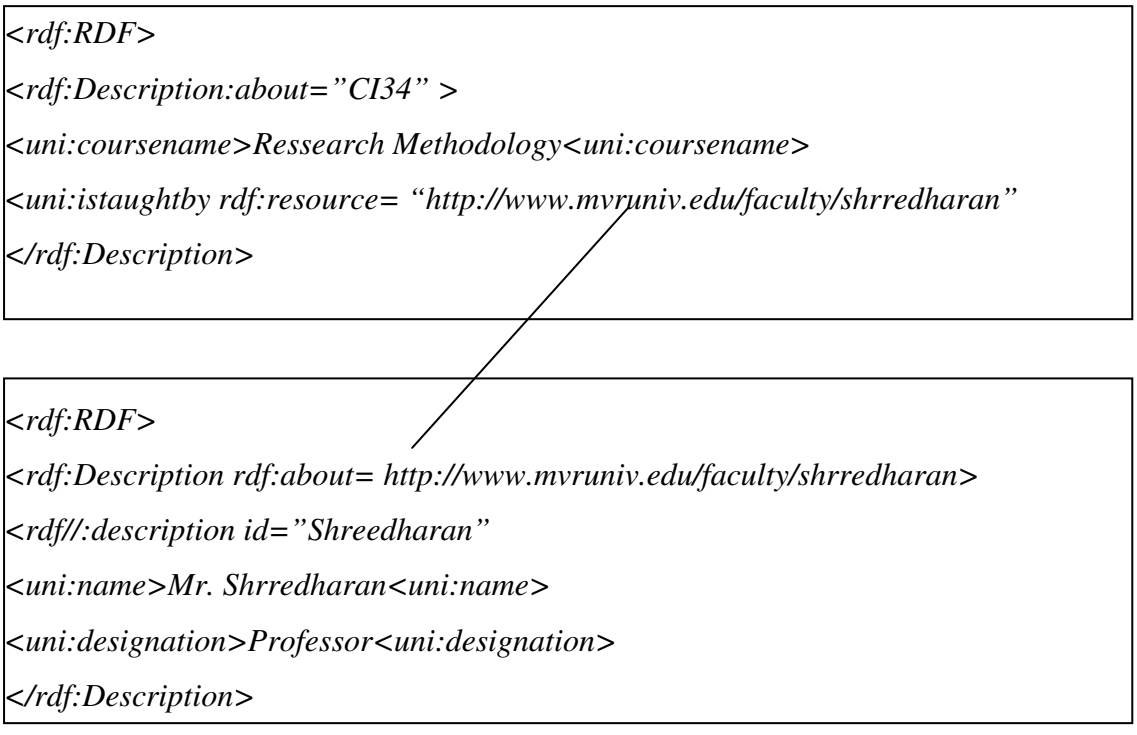

Figure 2.24 Hyper linking of RDF fragments

The example in Figure 2.24 indicates the referencing model of URI where URI are further used for crawling. The property *istaughtby* has a link which is described at a different location, showing the hyper linking of RDF documents. The Ontologies are

of distributed nature that is the RDF fragment may have same URI but may be present at different locations. Hence, after crawling, the RDF chunks of ontology are aggregated.

2. Ontology Classification

After crawling and aggregating, ontologies are classified to fit into a predefined category of classification. For this, traditional text classification techniques are used. The term used for defining concepts and relationships are considered as plain text. The classification is based on algorithms like Naïve Bayes, TF-IDF/Rocchio, k-Nearest Neighbours algorithm (KNN) and Probabilistic Indexing. The Rainbow Tool [108] was used for classification which is a text classification tool based on the above mentioned algorithms. The Rainbow Tool is trained using DMOZ directory of classification. The ontology is then classified based on DMOZ classification category. Ontologies are further visualized using GraphViz, which converts the ontology into visual representation.

3. Ontology Ranking

For ranking the ontologies retrieved, specific features of underlying knowledge representation of RDF model and URIs like different types of links, distance, weights are considered. Priorities are given to different types of relationships.

In this algorithm, apart from considering the rank of the referrer, the weight of the type of reference (relationship) is also taken into consideration. Table 2.9 shows the weights of hyperlinks considered while ranking [108].

Table 2.9 Weights of Hyperlinks

Priority(Weight)	**Relationship**	**Language specific**
1	Instantiation	rdf:type
2	Subclass	rdfs:subclass, daml:subClass
3	Domain/range	rdfs:domain, daml:range

Metrics for Ranking Ontologies in OntoKhoj –

O = the ontology whose rank is to be determined.

α = the number of ontologies referring O

β_i = the number of referrals from ontology O_i to O.

Ω_i = the total number of outgoing referrals from ontology O_i.

T = the weight of the reference

N =normalization factor.

The OntoRank, OR (O) is defined in equation 2.4 as

$$OR\ (O) = N * \sum_{l=1}^{\alpha} 1 / \Omega_1 * \sum_{j=1}^{\beta i} OR\ (O_i) * T_j \qquad (2.4)$$

The above mentioned metrics is used to find the ontology rank corresponding to the weight and the number of referrals. The Ontokhoj search engine is implemented in Java to crawl ontologies from the Semantic Web, aggregate the different chunks of ontologies distributed using URIs, classify the ontologies and then ranking the ontologies in order of their decreasing rank based on weights of links and referrals of ontologies. This system is useful in finding ontologies in their ranked order.

2.7.2 Swoogle search engine

Swoogle [111] is a crawler-based indexing and retrieval system for the Semantic Web. It extracts metadata for each discovered document, and computes relations between documents. Discovered documents are indexed by an information retrieval system which either use character N-Gram [112] or URIrefs [112] as keywords to find relevant documents and to compute the similarity among a set of documents.

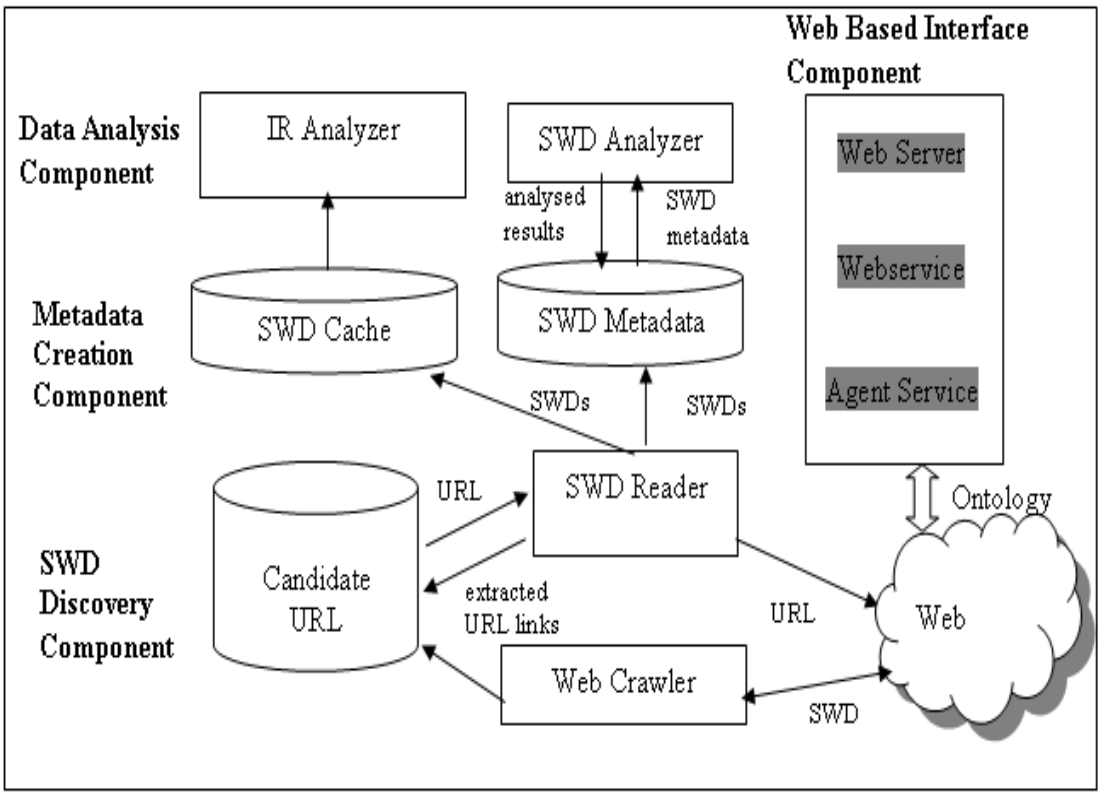

Figure 2.25 Architecture of Swoogle

The architecture of Swoogle consists of following main components:

1. SWD Discovery Component

This component has three different crawlers that discover Semantic Web documents on the World Wide Web. These crawlers are also responsible for keeping up-to date information about crawled semantic documents.

(i). **Google crawler** –Swoogle uses Google APIs to find Semantic Web documents, this is done by using Google web service "filetype", with this service files with extensions .rdf, .owl, .n3, .daml are searched and the set of documents returned act as seed URLs for Swoogle.

(ii). **Focussed crawler** –Based on the heuristic that, Semantic Web document found in one directory will probably contain more Semantic Web documents, the seed URL found with the Google crawler is fed to the focussed crawler to focus on a particular directory and crawl more Semantic Web documents linked to that directory.

(iii). **Swoogle crawler**-This crawler uses Jena framework for parsing and analyzes the contents of Semantic Web documents to discover more Semantic Web documents. This crawler works on the following analyzation of the contents-

a. *rdf:seeAlso* property of an instance links to another Semantic Web document
b. *owl: imports* links to another Semantic Web document
c. The construct *foaf: Person* in a FOAF ontology links to another FOAF document.

Therefore, for discovery of Semantic Web documents, Swoogle uses three crawlers Google crawler, Focussed Crawler, Swoogle crawler and in addition Swoogle has the web based interface to allow the registered uses to submit a URL of Semantic Web document, which is also used by Swoogle as seed URLs.

2. Metadata Creation Component

This component is used to collect semantic metadata to make the search experience more effective and efficient. This component collects three different types of metadata:

(i). Basic Metadata

Basic metadata of a Semantic Web document includes the following information

- **Encoding**- It gives the encoding information of the document i.e. whether the document is "RDF/XML","N-TRIPLE" or "N3".
- **Language**- It specifies the language used in Semantic Web document. The languages identified by Swoogle are "OWL","DAML+OIL","RDFS" and "RDF".
- **OWL species-** It specifies the language species used in the document. An OWL document is specified in three OWL species "OWL Full", "OWL-DL" and "OWL-Lite".

(ii). RDF statistics Metadata

This metadata information classifies a document either as Semantic Web Ontology (SWO) or an instance document which is called as Semantic Web Databases (SWDBs) based on ontology-ratio metrics as mentioned in [111,113] equation 2.5 as

$$R(foo) = \frac{|C(foo)|+|P(foo)|}{|C(foo)|+|P(foo)|+|I(foo)|} \quad (2.5)$$

Where for a Semantic Web document

|C (foo)|=refers to set of classes

|P (foo)|=refers to set of properties

|I (foo)|= refers to set of individuals

The value of R (foo) varies between 0 to 1, where "0" means pure SWDB and the value of ontology-ratio "1" is classified as pure "SWO".

(iii). Relational Metadata

This component collects information about the inter-relations that exists among different Semantic Web documents. These relations are identified by analyzing the Semantic Web document. The different inter-relations that exists between documents are-

- **IM**-One SWD can import another SWD. Constructs used for this are owl: imports
- **EX:** one SWD extends another SWD. Constructs used for this are *rdf:subPropertyOf*, *rdf:subClassOf*

3. Data Analysis Component

This component uses the information of the metadata component which classifies SWDB with SWO using ontology-ratio metrics to further calculate the rank of Semantic Web documents. The ranking algorithm of Swoogle is based on PageRank algorithm of Google, but the random surfer model of Google is not suitable for Semantic Web documents because of the different semantics of the underlying language used for representation for Semantic Web.

4. Interface component

This component deals with the different services to be provided to the user and Semantic Web community which are as-

- Web Server- Supports human user interface through the website http://swoogle.umbc.edu[114]
- Web Service- are provided using REST[111] interface
- Agent Service-Services to provide Swoogle as agent service.

Swoogle is a search engine for searching ontologies which provides the method for crawling ontologies, metadata creation, ranking of the ontologies.

2.7.3 Falcon

Falcon [115, 116] is a novel keyword based ontology search engine. It retrieves concepts whose textual description is matched with the terms in the keyword of the query and ranks the results according to both query relevance and popularity of concepts. The popularity is measured based on large datasets collected from the real Semantic Web. The architecture of Falcon Concept search [116] depicted in Figure 2.26 has been discussed as follows:

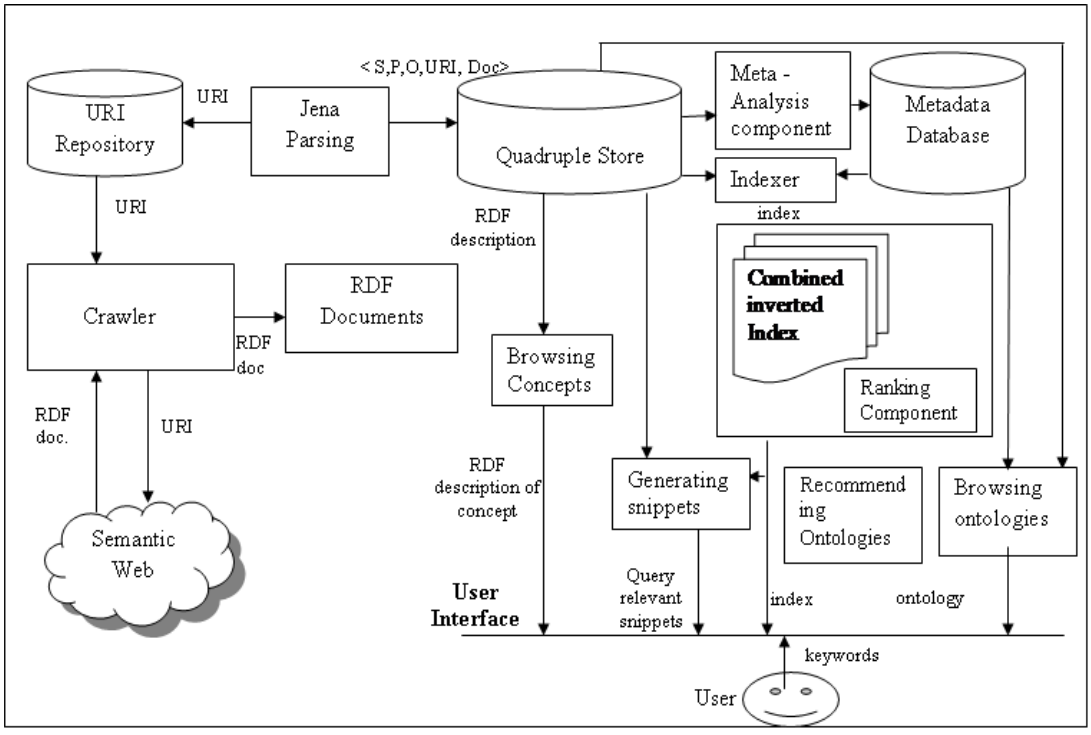

Figure 2.26 Architecture of Falcon Search Engine

1. Crawler

The Crawler application crawls the contents from the Semantic Web which are only Semantic Web documents using content negation which accepts only applications that are RDF/XML. The URIs in these fetched documents are sent to crawler for further fetching of the contents from the Semantic Web. The discovered contents are then parsed using Jena parser and submitted to quadruple store.

2. Quadruple store

In this component the parsed RDF triples from the RDF document and the document URI are stored in the quadruple store which is implemented based on MYSQL database.

3. Meta-analysis Component

This component analyses the semantic contents and creates metadata. It generates the metadata like the type of entity identified by a URI.

4. Indexer

The indexes are generated using Apache Lucene [117]. Unlike traditional model where indexes are created for terms to the URIs of the pages containing those terms, such an index cannot be created for Falcon as the concepts are RDF triple. In this system there are following two indexes generated:

(i). for each concepts, terms (which are extracted from RDF description), are linked to its virtual documents that contain those terms

(ii). Second index consists of ontologies to the concepts.

For each query entered the result set is based on the intersection of the result set obtained from these inverted indexes.

5. Ranking

This components ranks the concepts based on the

(i). Popularity of concept

(ii). Term based similarity between the virtual document of concepts and the user entered query.

The ranking metrics is given by equation 2.6 as

$$\text{RankingScore}(c, q) = \text{TextSim}(c, q).\ \text{Popularity}(c) \qquad 2.6$$

The raking score of a concept to a query entered is given as the product of textsim of a concept to a query and popularity coefficient of concept.

6. Generating snippets

This component is responsible generating query-relevant snippet for each concept from the data stored in the quadruple store.

7. **Recommending Ontologies**

This component recommends ontologies to the user based on the ranking concepts

8. Browsing concepts

This component returns the RDF description from the quadruple store of the concept queried by the user.

Falcon is a search engine for Semantic Web documents which parses the Semantic Web documents into RDF triples and allows the keyword based queries on objects, concepts and ontologies.

2.7.4 OntoSearch Search Engine

OntoSearch [181] is a kind of "Ontology Google", which help users find ontologies on the Internet. OntoSearch combines Google Web APIs with a hierarchy visualization technique. It allows the user to perform keyword searches on certain types of “ontology” files, and to visually inspect the files to check their relevance.

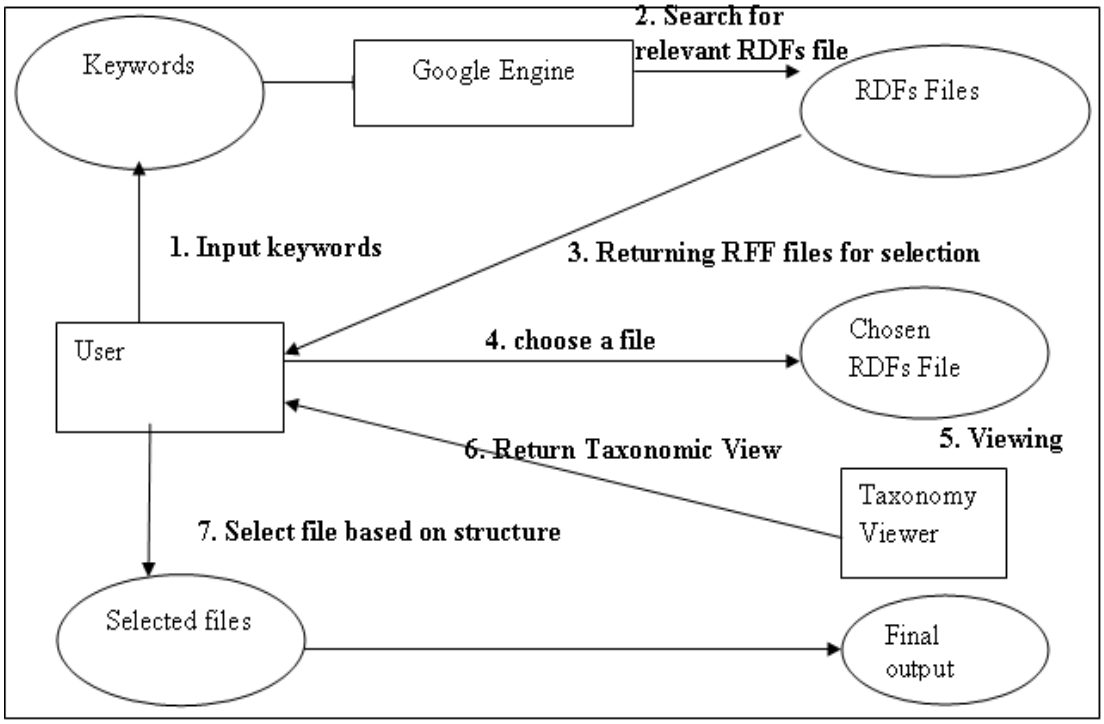

Figure 2.27 Flow of Ontosearch

Figure 2.27 depicts the flow of OntoSearch system which uses the Google Search engine for inputting the keywords and the relevant RDFs files are searched matching the keywords returning a set of RDF files, which are presented to the user for selection. User chooses the relevant RDF files from a set of returned results and displays the file in a hierarchical view which is returned to the user. Based on the structural analysis of ontology, user selects the files and save the selected ontology to the ontology library for further use.

OntoSearch system is implemented in Java and JSP and Jena APIs which is used for searching ontologies and maintaining an ontology repository to be used as web service.

2.8 SEMANTIC WEB QUERY LANGUAGES

Query languages used for querying from structured databases such as Structured Query language (SQL) and languages used for querying from web data such as XML Path Query Language (XPATH) cannot be used for querying semantic data specified in languages such as RDF, OWL. Among the various languages developed for querying semantic data content SPARQL [119] is the most commonly used query

language for Semantic Web as all others are languages lacks a lot of features and are not supported by many tools and hence are not widely used. The various query languages developed have been as discussed below:

1. SquishQL(SQL like Query language)[178]

Squish (pronounced as "SQL-ish) query syntax are similar to SQL query language. It is a query language based on graph navigation. SQL query language for RDF provides consistent, human-understandable, access to repositories of semantic data, whether stored files or large databases, enabling application programmers to create Semantic Web applications quickly [178]. For example consider Figure 2.28:

```
SELECT ?title
FROM http://example.com/xmleurope/presentations.rdf
Where
        (?doc, <dc:title>,? Title).
        (?doc, <rdf:type>,<foaf:Document>)
Using
        dc      FOR  <http://purl.org/dc/elements/1.1/>,
        foaf FOR  < http://xmlns.com/foaf/0.1>,
        rdf FOR <<http://www.w3.org/1999/02/22-rdf-syntax-ns#>
```

Figure 2.28 Query to find the titles of documents in http://example.com/xmleurope/presentations.rdf adapted from [178]

Figure 2.28 shows a query in SquishQL which selects title from *http://example.com/xmleurope/presentations.rdf* document, by selecting the document where the predicate is <*dc: title*>, and the documents are of type FOAF documents. *Using* clause specifies abbreviation for long URIs by defining a string prefix; this example specifies URIs for Dublin Core (DC), FOAF (friend-of-a-friend), RDF (Resource Description framework).

2. RDQL (RDF Query language) [179]

This language was developed by Hewlett Packard. The syntax of RDQL is similar to SQL select clause, but does not include *from* clause. Consider an example of RDQL query

Select ?x where(?x,<rdfs:label>,"abc")

This query will list all resources with *abc* in the variable *x*.

3. SeRQL (Sesame RDF Query Language) [122]

SeRQL (pronounced "circle") is considered as second generation RDF Query language. This language is based upon earlier query languages such as RDQL [179] and N3. SeRQL uses a path expression syntax [123] that is similar to the syntax used in RQL, and is based on the graph nature of RDF; the path is expressed as a collection of nodes and edges, where each node is denoted by surrounding curly brackets.

{node} edge {node} edge {node}

Consider an example to query, RDF graph for Book Title Published by *Pearson*, the path expression for this query would be specified by

{Title} <foo:publishedby> {Publisher} <rdf:type> {foo:Pearson>}

This query will list all Book titles by publisher *Pearson.*

4. SPARQL (SPARQL Protocol and RDF Query language) [118, 119]

SPARQL is a standard language for Semantic Web. It is W3C standard recommendation for querying RDF data content. SPARQL query language is based on matching of RDF triple Graph pattern. The different constructs supported for RDF query language are:

- SELECT Query
- CONSTRUCT Query
- DESCRIBE Query
- ASK Query

SPARQL currently has W3C Candidate Recommendation status as being the "Query Language for RDF". In particular, SPARQL has the following facilities [119]:

- Extract RDF subgraphs,

- Construct a new RDF graph using data from the input RDF graph queried,
- Return "descriptions" of the resources matching a query part,
- Specify optional triple or graph query patterns (i.e., data that should contribute to an answer if present in the data queried, but whose absence does not prevent an answer being returned),
- Test the absence, or non-existence, of tuples.

Similar to SQL, which is used for querying structured databases, SPARQL queries is used to query unstructured databases and have a SELECT-FROM-WHERE structure. There are other query languages which are considered as first generation query languages, which has a good expressive query constructs feature set but they are not supported by all the tools for ontology development and lacks interoperability feature, hence these query languages are not considered as the standard language for querying. For example, consider a semantic data fragment of an FOAF ontology which consist of name, designation and email-address and other information of a person. Query on such data to query name, designation with SPARQL can be used as shown in Figure 2.29.

```
PREFIX foaf:<http://xmlns.com/foaf/0.1/.
PREFIX exof: foaf:<http://example.org>
SELECT   ?name   ?designation
WHERE {
?x foaf: name    ?name
foaf: designation   ?designation
}
```

Figure 2.29 Example of SPARQL

Figure 2.29 shows example of SPARQL to select *name, designation* variable where predicate is *foaf: name and foaf: designation from http://example.org.* The main query constructs used in this query are as follows:

(i). PREFIX- PREFIX keyword is used to declare a namespace for a URI.

(ii). SELECT-This keyword is used to specify data items that will be included in the result set. In this for example variable name, designation is included in result set.

(iii). FROM- This keyword specifies the data set on which the query will be executed.

(iv). WHERE-This keyword specifies the triple/graph pattern which query will match against a RDF graph.

Variable names have question mark in their beginning. This triple query will be evaluated against all the triple that exist in the semantic data.

5. SQWRL (Semantic Query enhanced web rule Language) [180]

It also has SQL-like operations to query knowledgebase of OWL. It is considered as an expressive language for performing queries on OWL ontologies. SQWRL takes a standard SWRL rule antecedent and effectively treats it as a pattern specification for a query. It replaces the rule consequent with a retrieval specification. The core SQWRL operator is sqwrl: select [180]. Consider an example Query **"Return all the person whose age is greater than 18"** of Figure 2.30

Person(?p)^hasage(?p,?a)^swrlb: greaterthen(?a,18) →sqwrl: select(?p, ?a)

Figure 2.30 Example of SQWRL

The query in Figure 2.30 find all the person (*Person(?p)),*whose age(*hasage(?p, ?a)),* is greater then 18 *(greaterthen(?a,18)),*and lists all the person and their age *(sqwrl: select(?p, ?a)).*

2.9. SUMMARY OF VARIOUS ONTOLOGY TOOLS

The summarization of various ontology tools used in various aspects of lifecycle of ontology has been given in Table 2.10

Table 2.10 Summarization of various Ontology Tools

Tools	**Examples**
Ontology Editor Tools	1. Protege, 2.SWOOP 3. NeOn Toolkit 4. Apollo 5. WebODE 6.OilEd 7.OntoEdit 8.Topraid Composer 9.OntoStudio
Ontology Annotation Tools	1. Annotea
Ontology Reasoning Tools	1. Pellet 2. Jess 3. RacerPro 4. Fact++ 5. Kaon2
Ontology learning tools	1. Protégé with OntoLT 2. ODEMapster
Ontology evaluation tool	1. OntoAnalyzer 2. OntoClean 3. RaDON
Ontology Storage Framework	1. Redland 2. Sesame 3. AllegroGraph 4. Virtuoso
Ontology merging and Alignment Tools	1. Chimaera 2. PROMPT

Search engines for both the conventional web and the Semantic Web involve the same set of high-level tasks: discovering and revisiting online documents, processing user queries and ordering search results. The details diverge, due to the differences in the distribution of SWDs and the semantic of their content [124]. Existing approaches for the Semantic Search Engines has broadly the following features:

- Ontology design is a time consuming and complex process, so design of such search engines enables the concept of ontology reuse by finding the ontology for a particular domain
- The documents are represented in highly structured languages hence allow the meaning based linkages among the documents.

The existing search engines for example Google searches web documents, Semantic Search Engines like Swoogle searches OWL, RDF documents which are represented in highly structured languages whereas there is requirement of representing web pages to be with highly structured languages and allow the concepts to be associated to them and an information retrieval framework to be designed for searching such web pages.

The next chapter discusses about the architecture of **SemEngine.**

CHAPTER III

3. PROPOSED WORK FOR SEMENGINE: SEARCH SYSTEM FOR ONTOLOGY

3.1 INTRODUCTION

Today's web, where information is generally represented by unstructured languages, and interpretation\ identification of relevant information is left upon user to evaluate, is termed as syntactic web. Syntactic web refers to a web where computers do the presentation and people do linking and interpretations [87]. For the case of machines to understand the underlying concepts and produce precise and relevant results there is requirement to move from syntactic web to Semantic Web. Semantic Web is a vision proposed by Tim-Berner Lee, which defines the importance of web contents to be represented with knowledge based approach [10, 46]. Ontology is one of the knowledge based technique to represent web contents and web pages represented by such contents are called Semantic Web pages/documents. These web documents/pages are generally represented by Semantic Web languages like Resource Description Framework (RDF), Web Ontology Language (OWL), and Darpa Agent Markup Language (DAML).

A search engine is an information retrieval system that provides relevant web information to the users but it is not possible to use current search engines for searching Semantic Web documents for the following mentioned reasons [125]:

a) Current techniques do not allow to index and retrieve semantic tags
b) They don't use the meaning of tags
c) Can't display results in visual form
d) Ontologies are not separated entities which usually have cross references that current engines don't process.

Thus, the movement from syntactic web to Semantic Web requires design of search engines which provide the results based on inference of the knowledge contained in

the web pages. The Semantic Web can be made a reality by gradually augmenting the existing data by annotations. Annotating the web contents with ontologies can be used as a solution to represent the knowledge on the web. Annotations can be either stored in the very same document, in an external repository, or they can be generated on the fly using lightweight human language technology [126].

The search engine in the traditional web follows three step processes: crawling, indexing, and searching. The proposed approach, following these basic steps, is different from the traditional model as it involves ontology annotated documents with crawling, indexing and concepts search on these documents. Traditional web search engines [9,127] have the limitation that web search results are based on keywords and not on concepts. Hence, various language issue like word sense disambiguation exists, for example

- Homonyms (same words representing different concepts) e.g. the "jaguar" car vs. "jaguar" animal
- Synonyms (different words representing the same concept) e.g. "car" and "automobile" represent the same concept and result set should include pages of both the concepts.

In computer science, the process of mapping terms to concept-space is referred to as Word Sense Disambiguation (WSD) [128]. In general, there are two main approaches to WSD; supervised and unsupervised approaches [129]. Supervised approaches [129,130] uses machine-learning techniques that learn to classify senses from examples (i.e. training sets) whereas unsupervised approaches [129] do not depend on training sets but uses techniques to utilise the information provided in the applied corpus (e.g. word collocation, keywords and part-of-speech) which are called as knowledge based approaches using ontologies. Enriching documents with ontologies are a knowledge based approach with machine understandable mark-up that formally represents a universe of discourse by describing the relationships between its concepts [70,128].

The problems of imprecise and irrelevant results continue to hinder web searchers, especially with the continued expansion of the web [131]. The example for such

imprecise results is as shown with the snapshots from Google search engine with query “Laptop Specification” in Figure 3.1.

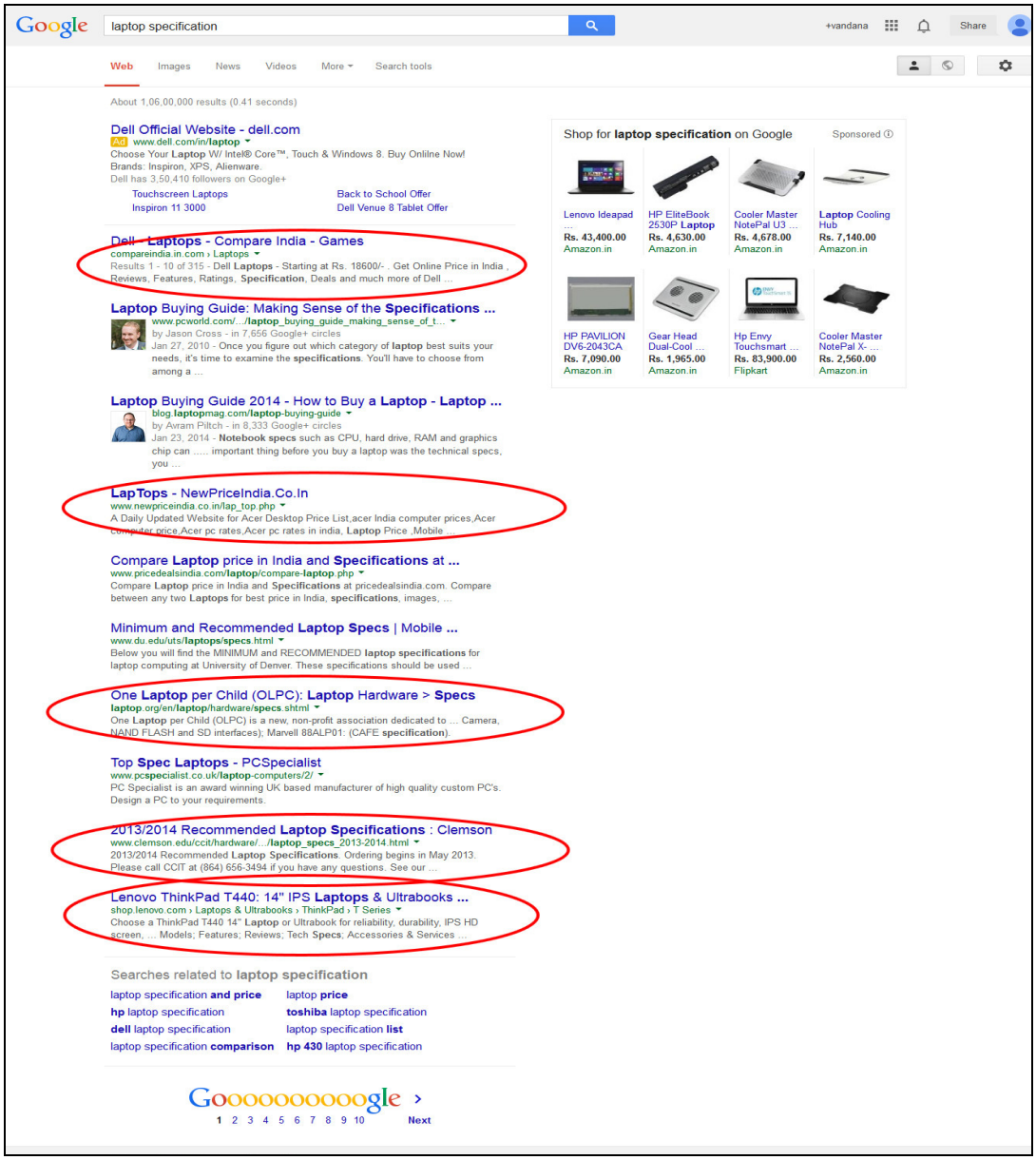

Figure 3.1 Google result for query “Laptop Specification”

Figure 3.1 shows the results of Google for the specified query “Laptop Specification”, where user intention to find the result consisting of specifications of laptop yields a result set which either specifies a specific site from where laptop can be purchased or is more specific to a brand. Only five out of first ten result set consist of results which are relevant results, giving only 50% performance of searching.

The approach followed in the research carried out can be described using a four step process:

Step 1. Crawling of the web pages, filtering out the web pages which are not annotated with any ontology,

Step 2. Indexing of the crawled annotated web pages,

Step 3. Ranking of the ontology annotated web pages.

Step 4. Searching of the web pages based on the concepts given in the query.

The following Table (see Table 3.1) describes comparative analysis of various search engines, done based on the approaches used and their input and output result format:

Table 3.1 Comparative analysis of Search Engines

Search engine	Approaches Used	Output Result Format	Input Format	Techniques Used
Hakia [132]	It is based on producing results based on searching of structured text like Wikipedia.	HTML web pages	Natural Language questions or phrases, keywords.	Proprietary semantic technology called QDEXing(Query Detection and Extraction)
DuckDuckGo[133]	It produces results based on searching of Wikipedia, Wolform Alpha.	Classified results with their HTML web pages giving the possible meaning for the query entered.	Natural Language	Clustered approach and NLP techniques
Cognition[134]	It produces results based on ontology and WordNet vocabulary.	HTML link results	Natural language phrases	Uses - Linguistic technology -Boolean search -Fuzzy search technologies to produce result.
SenseBot[135]	Concept search	Summarized results	Query using keywords	Uses text mining algorithms that parse the web pages to produce results.
PowerSet[136]	Based on giving results searching the contents of Wikipedia.	HTML web pages	Query using keywords, natural language questions or phrases	Powerset semantic indexing is based on the XLE (Xerox Linguistic Environment), Natural Language Processing technology.
Google[42]	Based on giving results based on keywords present in pages.	HTML web pages	Natural Language	PageRank Algorithm

Swoogle[112]	Have ontology repository.	Finds appropriate ontologies and list them in ranked order.	Domain concepts	-N gram based indexing -Ontology rank based on PageRank
Watson[137]	Find ontologies by integrating the search capabilities.	Ontology Listing	Domain Concepts	-Watson Semantic gateway -NeOn Toolkit
Falcon[116]	Concept search	Produces Ontology listing and generates query relevant structured snippets.	Keywords	-Popularity based approach for ranking of concepts and ontologies

Table 3.1 shows the comparative analysis of various search engines done on the basis of approaches used, output result format and input format used by these search engines. Most of the search engine described above search on the basis of keyword matching like Google, whereas some of them are called as semantic search engines but are generally used for finding ontology which can be reused for a particular domain. The examples of such search engines are Swoogle [112], Watson [137], Falcon [116] etc.

This chapter describes the proposed framework of "SemEngine: Search System For Ontology", for searching ontology annotated documents describing the methodology followed for developing the system depicting the functional diagram of the developed system, and describing the implementation and the experiments done for evaluation. The technique followed by SemEngine is compared with various search engines on different parameters shows that proposed SemEngine is both efficient and scalable.

3.2 PROPOSED FRAMEWORK FOR SEMENGINE: "SEARCH SYSTEM FOR ONTOLOGY"

The proposed SemEngine's functionality includes

- Deploying the strategy for crawling the ontology annotated web pages,
- Indexing ontology annotated web pages,

- Ranking the web pages associated with ontology in order of their relevancy with SemRank algorithm,
- Searching the query against matching ontology annotated web pages giving a ranked set of web page results.

The architecture of the SemEngine has been proposed that uses a novel technique to search ontology annotated web pages. The proposed architecture consists of the following functional components:

1. Ontology Development Framework
2. SemCrawl Module
3. SemIndex Module
4. SemRank Module
5. Query Interface Module
6. Search Module

Each component of proposed SemEngine has been discussed in brief in this chapter and details of each component with implementation are discussed in next chapters. The description of each module of proposed SemEngine with their proposed algorithm is described below:

3.2.1 Ontology Development Framework

Large numbers of development frameworks are available for Ontology Engineering like Protégé [61, 62], OntoStudio [138], SWOOP [99]. Among these Protégé is most widely used by researchers, which is developed by Stanford University and is an open source tool. In Protégé concepts, properties like data properties and properties between concepts can be easily modelled. The consistency of the developed concepts can be checked by different reasoners available, different plug-ins reasoners available for Protégé framework are Pellet [101], Jess [139]. Query retrieval can be done in Protégé framework using DL (Description Logic) Query [140], rule languages can also be used with Protégé development framework like SWRL (Semantic Web Rule Language) [107].

The proposed architecture is described in Figure 3.2.

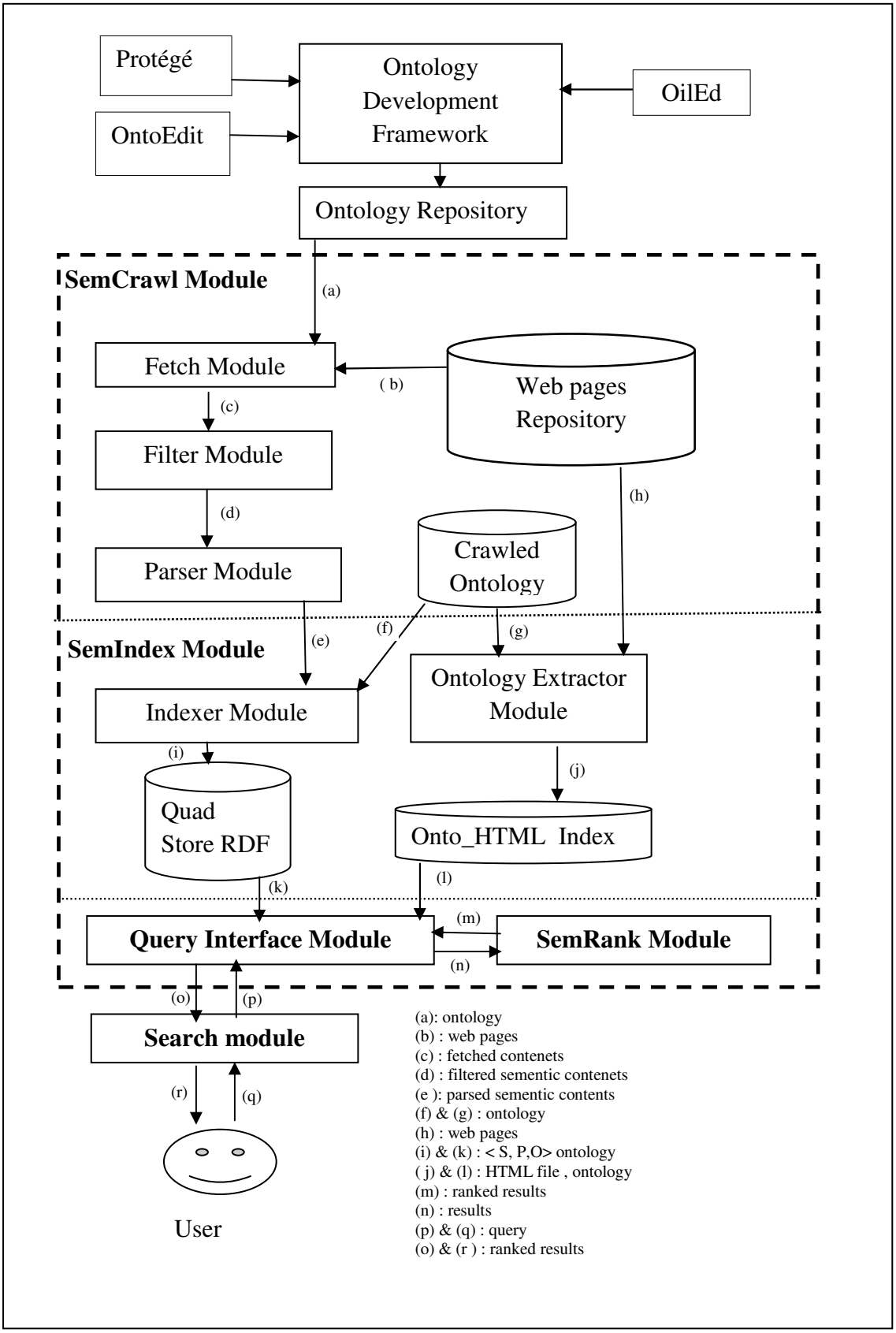

Figure 3.2 Proposed Architecture for Search System for Ontology Annotated Web Documents

In the current research, Protégé development framework has been used for development of ontologies in laptop domain. The three ontologies have been developed in laptop domain i.e *Laptop_Specification, Laptop_Seller* and *Laptop_ Review* ontology consisting of specific concepts and properties related to it. The ontologies have been developed using Protégé tool. Fact++ Reasoner [141] has been used to check the consistency of the concepts used. DL Query has been used for validation and of concepts related to ontology [142].

The details of this module have been given in **Ontology Development in the Domain of Laptop (Chapter 4).**

3.2.2 SemCrawl Module

This is an important module that crawl the web pages, domain ontologies that are used for annotation of web page, extracts the semantic contents described in various semantic web languages. These crawled web pages are then further filtered and parsed. The output of the SemCrawl Module is given as input to SemIndex module.

This proposed SemCrawl module [48] is mainly responsible for crawling of the web pages annotated with domain ontologies from Web Page Repository and the associated ontology from Ontology Repository. The web pages which are semantically annotated are given to indexing module whereas the pages without annotation will be filtered out. The crawled semantic ontology is parsed by the Parser Module where each statement is parsed into <S, P, O> where S represents subject, P represents predicate and O represents object. The output of SemCrawl Module are the filtered annotated web pages which are parsed as <Subject, Predicate, Object > and are given as input to the SemIndex Module for further indexing purpose.

Fetch Module upon receiving the *contentempty* signal from the Filter Module, extracts URI from URI Queue and fetches the contents from the web based on URI link received. Upon receiving the contents from the web, it stores the contents in the semantic buffer and signals *contentready* signal to the Filter Module signalling, the availability of the contents.

The algorithm for modules of SemCrawl Module is summarized in Figure 3.3.

```
Algorithm SemCrawl ( )
{
Fetch Module ( );
Filter Module ( );
Parser Module ( );
}
Fetch Module
{
        wait (contentempty);
        extract URI from the URI Queue ;
        fetch the contents from corresponding URI;
        store the contents in semantic buffer;
        signal (contentready);
}
Filter Module
{
        Wait (contentready);
        get  the contents from buffer;
        extract the semantic  web pages;
        send the filtered contents for parsing;
        signal (contentempty);
}
Parser Module
{
        extract the crawled ontology;
        extract subject, predicate and object;
        store triples in database with three columns subject, predicate,
              object<S,P,O>;
}
```

Figure 3.3 Pseudo code for SemCrawl Module

Filter Module upon receiving the *contentready* signal from the Fetch Module, inputs the contents from semantic buffer and filter the web pages associated with ontology and send the filtered web contents for further extraction of link from the contents, and send the signal *contentempty* to Fetch Module for further fetching of contents.

Parser Module receives the crawled contents from semantic repository and extracts subject, predicate and object from crawler output repositories and stores those triples in database.

The output of SemCrawl Module; crawled semantic web pages and ontologies are given as input to the SemIndex Module.

The details of this module have been discussed in **SemCrawl: A Framework for Crawling Ontology Annotated Web Documents (Chapter 5).**

3.2.3 SemIndex Module

The SemIndex Module [143] creates the indexes for crawled web contents for efficient retrieval. The Indexer Module creates one of the indexes; <Subject, Predicate, Object, Ontology> example of the index is depicted in Table 3.2. The other index is created by Onto_HTML Mapping Module; <HTML, Ontology>. The two indexes created by SemIndex Module are

- <Subject, Predicate, Object, Ontology>,
- <HTML, Ontology>

The example of <Subject, Predicate, Object, Ontology> index is depicted in Table 3.2.

Table 3.2 Index of <Subject, Predicate, Object, Ontology>

Subject	**Predicate**	**Object**	**Ontology**
RAM_4G	Type-of	Laptop_Memory	Laptop_specification Ontology

This example in Table 3.2 shows that *RAM_4G* is a type of *Laptop_Memory* and this is associated with ontology *Laptop_Specification* ontology. Other index generated by index module is <HTML, Ontology>. The example of <HTML file, Ontology> is depicted in Table 3.3.

Table 3.3 Index of <HTML file, Ontology>

HTML File	**Ontology**
E:\projects\RDF\Final\crawled-data\filtered-web-content\RAM_Specs.htm	Laptop_specification Ontology

A snapshot of the Onto_HTML Mapping Database Module is depicted in Figure 3.4.

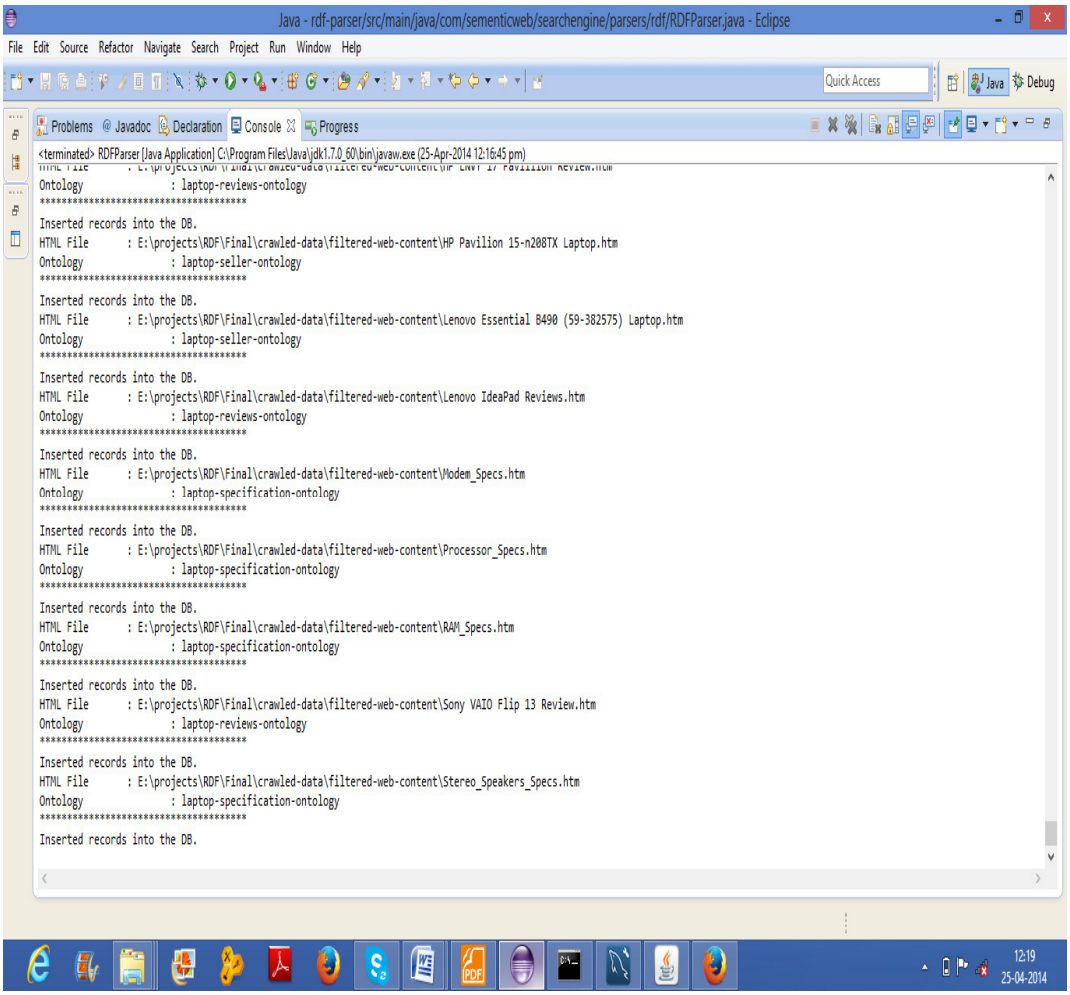

Figure 3.4 Parsing of results to <HTML, Ontology>

The example shown in Figure 3.4 indicates one of the example of <HTML File, Ontology > mapping index of Onto_HTML Mapping Database Module.

HTML File : E:\projects\RDF\Final\crawled-data\filtered-web-content\snapdeal.htm
Ontology : laptop-seller-ontology

Figure 3.5 Ontology and the web page associated with that ontology

The example in Figure 3.5 shows that HTML web page from *www.snapdeal.com* website giving information about laptop price is associated with *Laptop_Seller* ontology.

The algorithm for SemIndex Module is depicted in Figure 3.6.

```
Algorithm SemIndex  Module ( )
 {
While (empty (repository)) do
{
        Indexer ( );
        Ontology Extractor ( );
}
}
Algorithm Indexer Module(input: parsed output, output: index)
 {
        read ontology file from the crawled  ontology  repository;
        parse ontology file ;
        extract RDF triples;
        insert subject, predicate, object ,ontology value to quad store RDF index;
}
Algorithm Ontology Extractor (input: web pages, output: index)
 {
        parse web pages from Web Page Repository;
        parse ontology from crawled ontology repository;
        extract the ontology name and associated HTML web page;
        index (HTML File, Ontology) in the database;
}
```

Figure 3.6 Algorithm for SemIndex Module

Indexer Module reads ontology file from Ontology Repository and parses ontology into triples and insert those triples and ontology value to database. The other module OntoHTML Mapping Module reads RDF files from repository and creates index of <HTML File, Ontology> onto the database.

Details of this module have been discussed in detail in **SemIndex: Efficient Indexing Mechanism for Ontologies (Chapter 6).**

3.2.4 RankEngine Module

The ranking module first selects the ontologies that are matched by the query entered by user, then it ranks all the pages that are annotated by selected ontology. The proposed algorithm consists of a hybrid approach having two dimensions of ranking; ranking metrics for selecting ontology and metrics for ranking web pages associated with the selected ontology using traditional metrics.

a) First dimension of ranking metrics for selecting ontology

For selecting ontology the measure, *Match_Measure* is given as specified in equation 7.1 as

) ≤ ≤ [.(_ (

(7.1)

where

i= number of ontologies ranging from 1 to n

f (R) = function of Relatedness

α , β = parameters where $\alpha + \beta = 1$ & $\alpha \leq 1$; $\beta \leq 1$

Concept_Weight_Measure is given as specified in equation 7.2 as

$$Concept\_Weight\_Measure = \sum_{i=1}^{n} f(M, \mathrm{Cn}) \qquad (7.2)$$

where

n refers to number of concept in query

f(M, C_n) = function which gives value 1 if ontology contains a exact class label matching C_n and 0 if ontology does not contain a class label matching C_n.

b) Second dimension of ranking metrics

After selecting the ontology that has the maximum Match_Measure, the web pages associated with the selected ontology are ranked based on Rank_Score metrics for a query as defined in equation 7.5 as

Rank-score= α TextFreq(c, q) + β (Linkweight) (7.5)

Where

TextFreq(c, q) = frequency of the concept similar to the query in web page.

Linkweight PR (A) = weight measure of links from a web page given a document A.

$\alpha + \beta = 1$; $\alpha \leq 1$ and $\beta \leq 1$;

These above mentioned measures are used to find the web pages that match those concepts defined in the query with the web page. The details of this module have been discussed in **SemRank: A Novel Hybrid Approach for Ranking Ontology Annotated Web Documents (Chapter 7).**

3.2.5 Query Interface Module

The Query Interface Module consist of an interface so that query can be fired by the user for search. The query is processed as set of concepts. For the query "Laptop Specification RAM" the concepts are as described in Figure 3.7.

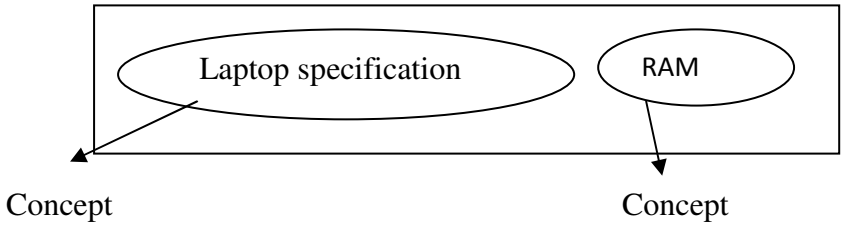

Figure 3.7 Concept matching of Query

Figure 3.7 shows the matching of the concepts, where the concepts "RAM" and "laptop specification" will be searched in the index for retrieval of results.

3.2.6 Search Module

The Search Module checks for the concepts associated in ontology, and matches on the basis of concepts contained in the ontology; the result is a set of URLs which are associated with those particular concepts. The algorithm for Search Module is described in Figure 3.8.

```
Algorithm Search Module (input: search term, output: ranked result)
{
        Input (search term)
        If search term(concept in ontology)then
                While (all urls are listed) do
                {
                List the corresponding web page to that ontology;
                }
}
```

Figure 3.8 Pseudo code for Search Module

The input for Search Module is HTML Mapping Index and output is the list of URLs, the search term is given as input through query interface, if search term is contained in ontology then until URLs are listed, all the corresponding web pages to that ontology are listed.

3.3 METHODOLOGY OF THE DEVELPOED SYSTEM

The step wise details of the implementation carried out to achieve the proposed research objectives have been depicted in Figure 3.10.

Step 1: Development of Ontology in a particular domain

In the first step three ontologies were developed in the domain of laptop. For the present research Protégé software tool is used for ontology development which is an open source tool. An overview of the developed ontologies is depicted in Figure 3.9.

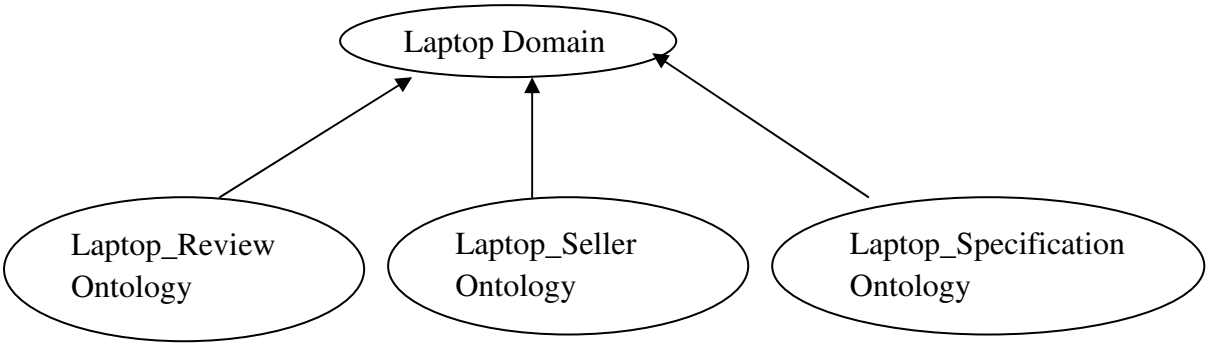

Figure 3.9 Three ontologies developed in Laptop domain

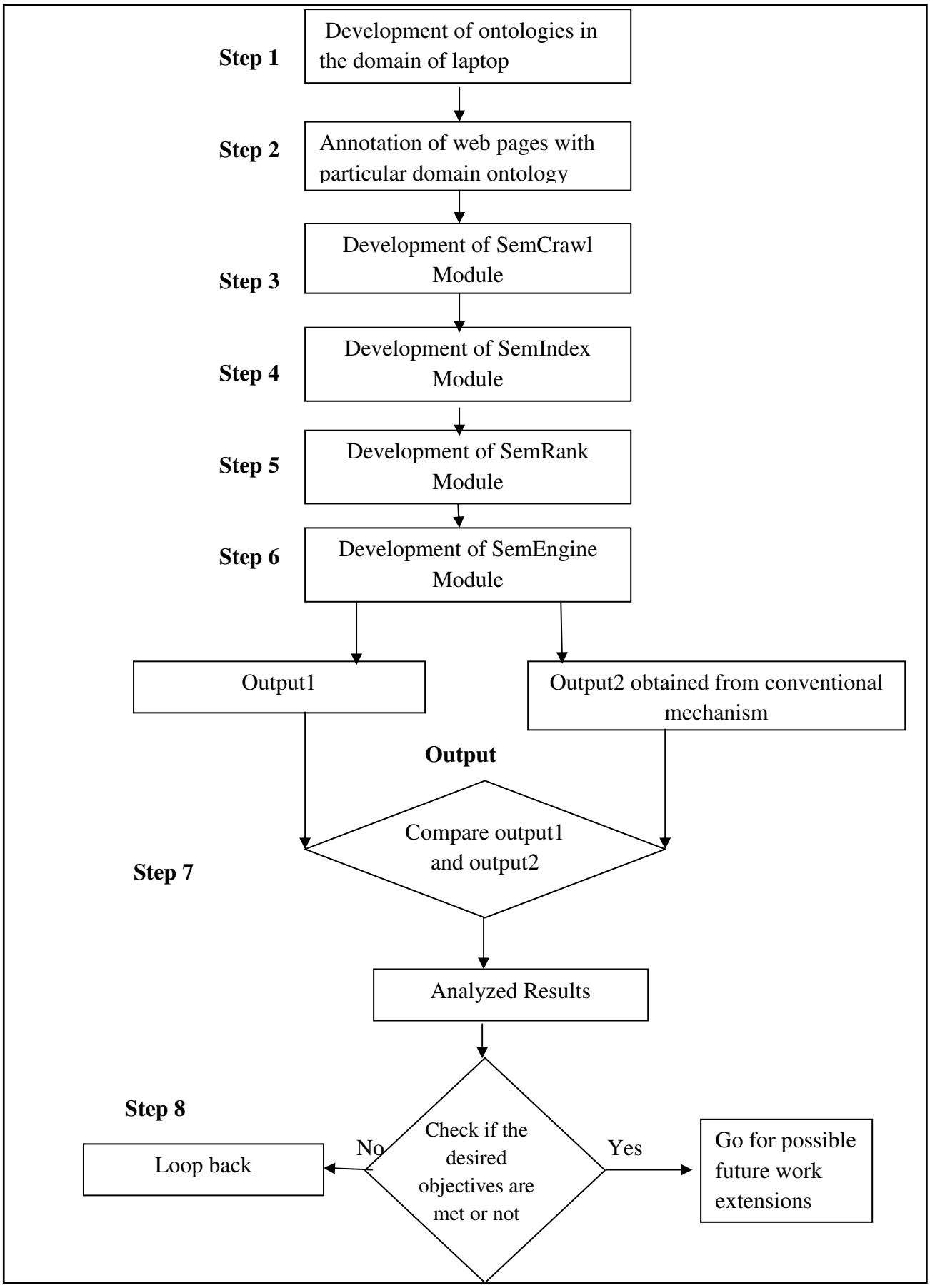

Figure 3.10 Methodology of developed system

The description of ontological framework shown in Figure 3.9 is as follows:

1. Laptop_Review Ontology

This ontology deals with the reviews relating to laptop. It covers the concepts like feedback, rating, reviews and other such concepts.

2. Laptop_Seller ontology

This ontology deals with selling sites of laptop. It covers the details like laptop_delivery_time, laptop_develivery_charges, laptop_payment_methods, laptop_availability, laptop_price and generalization to these concepts.

3. Laptop_Specification ontology

This ontology deals with specification concepts of laptop. It covers the concepts like laptop_os, laptop_memory, laptop_display_size, laptop_dimension_specs and more generalization to these concepts. The subclass for the concepts and instances for concepts are defined for example for laptop_dimension the subclass is laptop_height and laptop_width.

Step 2: Annotation of web pages with particular ontology

The web pages covering a particular scope is annotated with one of these developed ontologies. For example web page *http://www.blog.laptopmag.com/laptop-buying-guide.htm* is annotated with ontology which covers the scope of this web page with *Laptop_ Specification* Ontology.

Step 3: Development of SemCrawl Module

The SemCrawl Module fetches the web pages and filters the semantic web contents for further processing, and parses the contents into triples <S, P, O>. This module has the feature of extracting heterogeneous documents from the web which can then be indexed. This will help in the semantic retrieval of the information.

Step 4: Development of SemIndex Module

The SemIndex module indexes the web pages annotated with ontologies. The module indexes the parsed triples into 2 indexes

1. <S, P, O, ontology>

2. <Ontology, html file>

In the index <S, P, O, ontology> S stands for the Subject, P stands for Predicate and O stands for Object and ontology stands for the particular ontology associated with the web page.

Step 5: Development of SemRank Module

This module initially selects the required ontology and then list the web pages annotated with that ontology. The module works on two dimensions-firstly searching for an appropriate ontology and secondly ranking the web pages associated with that ontology. The output of this module is the ranking of the web pages as per the Rank-measure and Concept-Measure.

Step 6: Development of SemEngine Module

The SemEngine module checks for the concepts associated in ontology, and matches on the basis of concepts involved in the ontology; the output will be a set of URLs associated with those concepts. It includes a user interface which allows the user to enter the user query which will match the concepts in the index and lists all the web pages related to those concepts.

Step 7: Compare output1 and output2

Comparison of output obtained from SemEngine module is done with output obtained from conventional search engines and the results are analyzed for the input queries.

Step 8: Check if the desired objectives are met

The results of the developed system is compared with conventional system, if the objectives are met, the possible future extensions of the work can be carried out otherwise the system need to be redeveloped with different perspectives.

3.4 FUNCTIONAL DIAGRAM OF THE PROPOSED SYSTEM

The functional architecture of the proposed system as two phase diagram is given in Figure 3.11.

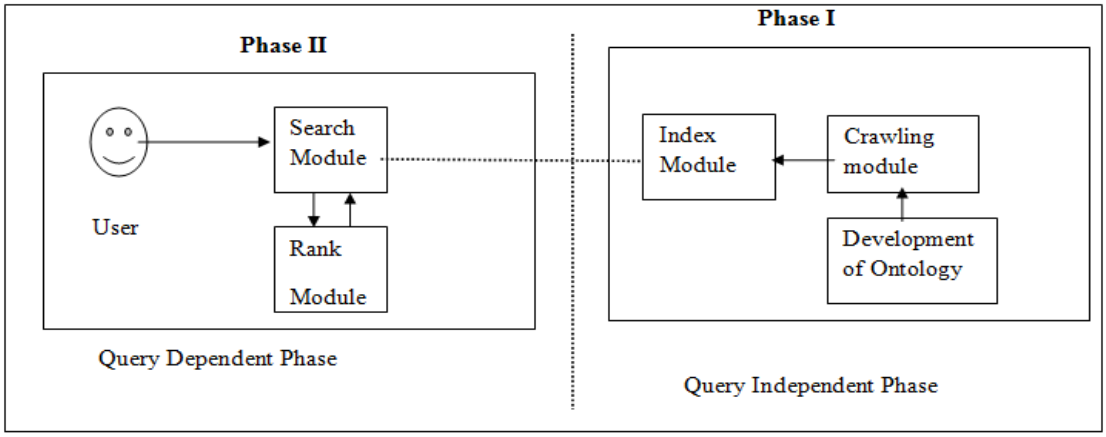

Figure 3.11 Functional Diagram of the developed system

Figure 3.11 depicts the two phase development of the system where Phase I is query independent phase and Phase II depicts query dependent phase. The query dependent phase includes the searching and matching of the concepts in the query with the developed index and ranking those pages in the decreasing order of their relevancy. Query independent phase includes the development of ontology using Protégé development tool, crawling those annotated documents and maintaining an index of the crawled corpus.

In this chapter, the proposed framework SemEngine has been discussed. This chapter discusses architecture of searching unstructured web pages which are annotated with the knowledge representation techniques called ontologies. The ontologies are well represented with Semantic Web languages RDF, OWL which can be created using various open source and commercial tools like Protégé, Altova Semantic Works. The annotation of web pages with ontologies helps machine to understand the semantics of the information represented and henceforth results into a more accurate retrieval of results for a query. The implementation of the proposed approach in the forthcoming chapters indicates that web information can be represented well with ontologies leading to better information retrieval. The research carried out envisions an approach of manually annotating web pages, which can be extended in future to create automatic annotation approaches to annotate the web pages which can be then crawled, indexed, searched and ranked using SemEngine architecture discussed in this research.

CHAPTER IV

4. ONTOLOGY DEVELOPMENT IN THE DOMAIN OF LAPTOP

Ontologies are the most important tool in knowledge representation, as they allow to logically relating large amount of data [144]. Ontology in general defines a common vocabulary to share information in a domain. The definition of ontology is as "An Ontology is a formal explicit specification of a shared conceptualization" [70, 71, 72]. Here, '*Formal*' refers to ontology being machine-readable. '*Explicit*' refers to the types of concepts, relationships between those concepts and constraints being explicitly defined. '*Shared*' refers to the concept that knowledge contained in ontology is consensual and has been accepted by a group of people. '*Conceptualization*' refers to an abstract model of phenomenon in the real world where the relevant concepts of that phenomenon has been identified.

The reasons identified for developing ontology are [75]:

1. To share common understanding of the structure of information among people or software agents.
2. To enable reuse of domain knowledge.
3. To make domain assumptions explicit.
4. To separate domain knowledge from operational knowledge.
5. To analyze the knowledge.

Ontology is generally specified using concepts called as classes, properties (slots) of concepts specified using object properties and data properties, individuals specified using instances of a class, and restriction on properties(facets) using slot cardinality, domain and range of a slot. Object properties exist between different classes and instances. Data properties exist between a class or instance and a data value.

4.1 ONTOLOGY DEVELOPMENT PROCESS

The process of Ontology development is not a linear process rather it is an iterative process which requires the revision and refinement of concepts for the evolving

ontology. Consider the development of *University* ontology that explains each step depicting the ontology lifecycle [10, 75].

1. Determine the Domain and Scope of Ontology

The first step in the development of ontology involves determining the domain and scope of ontology. The various things to be kept in mind while designing ontology under this step are as follows:

1. Domain where the ontology design is to be applied
2. Application of the ontology
3. The characteristics of an ontology
4. Type of application the ontology can be applied to
5. The type of questions the ontology should be able to answer
6. User of ontology and maintenance of ontology
7. Languages to be used which will be appropriately mapped to the intended application

For example, while designing ontology for *University*,

(i). Domain is "*University*"

(ii). Scope covered is "*Academic Teaching*". This ontology design describes the entities in academic domain of a *University*. Ontology scope refers to defining

- relation between student and the courses that student is pursuing,
- faculty teaching a particular course,
- courses in each department.

2. Considering the Reuse of Existing Ontology

Developing ontology from scratch is considered as a very difficult, time consuming process and requires a lot of domain knowledge, so it is always advisable to reuse an already existing ontology and extend with own concepts to meet one's requirement.

The following things must be kept in mind while considering the reuse of ontology:

1. Applications which can use the developed ontology for consideration of reuse,

2. Library from where ontology can also be reused rather than starting from scratch e.g. DAML Library [75,145], Ontolingua Library [75,146] has a large collection of ontologies.

For current research, there was no existing ontology in the domain of laptop that is meeting the specified requirement; therefore, the ontologies have been designed from scratch for the proposed research work.

3. Enumerate the Important Terms in the Ontology

This step involve finding out all the important terms that are expected to appear in ontology design. Generally nouns are considered as classes names and verbs form the property names. For example, while designing ontology for *University* the important terms are

- University
- Faculty
- Assistant professor
- Associate professor
- Professor
- Student
- Course
- Technology
- Computer Engineering

4. Define the Class and the Class Hierarchy

The different approaches for developing Class Hierarchy are [75]:

1. Top-Down- In this approach the development starts with the most general concepts in domain and subsequent specializations of concepts.
2. Bottom-up- This approach starts with most specific classes and then grouping of theses specific classes into more general concepts.
3. Combined Approach- This approach is combination of both the approaches and starts with the defining of more salient concepts and then generalizes and specialize them appropriately.

The different steps followed for developing *University* Ontology are -

1. Define the classes of *University* Ontology. Classes for *University* domain are *University, Department*, *Student*, *Course*, *Faculty* etc.
2. Define the hierarchy of the classes i.e. defining classes and subclasses of ontology design. The Class Hierarchy for *University* Ontology is depicted in Figure 4.1.

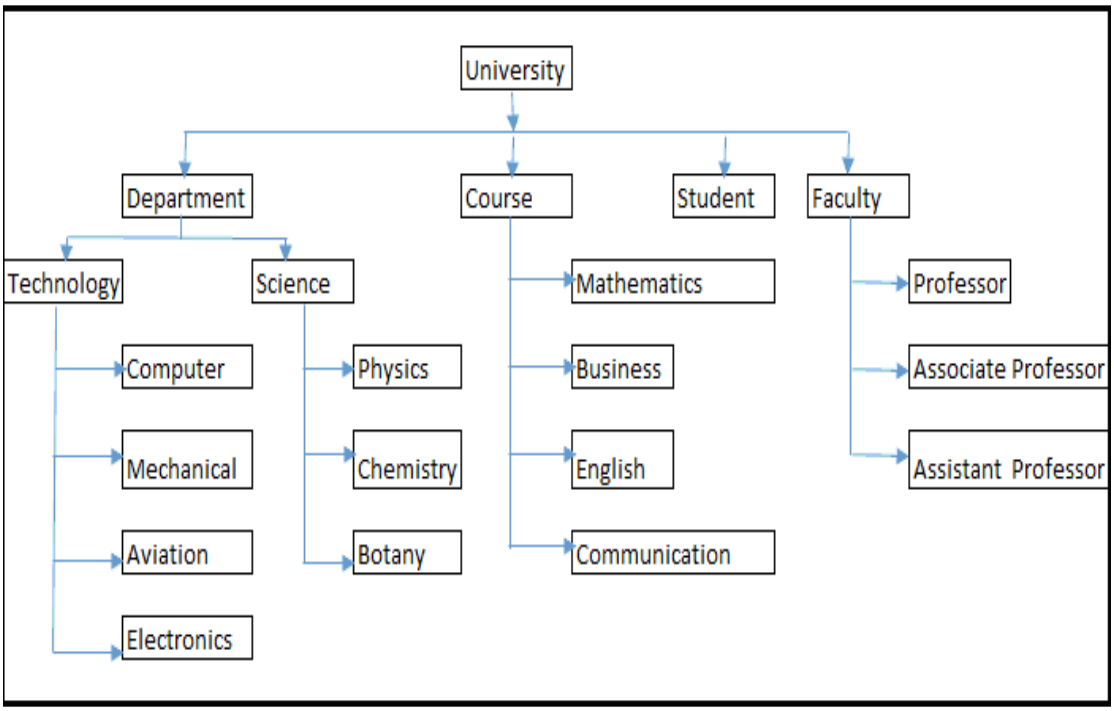

Figure 4.1 Class Hierarchy for *University* Ontology

Figure 4.1 shows Class Hierarchy where *University* is a super class and has *Department*, *Course, Student* and *Faculty* as subclass. *Department* is further classified into *Technology* and *Science*. *Faculty* is further classified into *Professor*, *Associate Professor* and *Assistant Professor*. This way of organizing classes into hierarchical taxonomy is called Class Hierarchy.

5. Define the Properties of Class-Slots

In this step, the properties that exist between different class i.e data properties define the relation between a class and value of a class and object properties which define the relation between two classes has been defined.

For example, for *University* Ontology *properties* can be defined as:

(i). Object Property -The different object *properties* that exists between classes has been defined as :

- *isTaughtby*

The *property isTaughtby* exist between two classes i.e *Course* and *Faculty*.

- *teaches*

The *property teaches* exist between two classes i.e *Faculty* and *Course*.

- *isEnrolledIn*

The *property isenrolledIn* exist between two classes i.e *Student* and *Department*.

- *hasSupervisor*

The *property hasSupervisor* exist between two classes i.e *Student* and *Faculty*.

(ii). Data Property- The properties that exist between different class, instance and data value are defined as:

- *hasName*

The *property hasName* exist between a class or instance and data value i.e class *Student* and name value of datatype string.
For example, Student *hasName* Pulkit

- *hasAge*

The *property hasAge* exist between a class or instance and data value i.e. class *Student* and value of datatype numeric.
For example, Akriti *hasAge* 32.

6. Define the Facets of the Slots

Slots have different facets which are mentioned by defining

1. Slot cardinality-It defines the number of value a slot can have. A slot can have single cardinality which means having at most one value and multiple cardinality means having any number of values. For example, student can belong to one department at a time so object property *isStudentof* is a single cardinality relation that exists between *Student* and *Department*.

2. Slot-value type-It defines the value a slot can take. It can be of different data types string, number, Boolean, enumerated data type e.g value of age is numeric.
3. Domain of slots-It refers to defining the domain and range of a slot.

Therefore, for the defined properties, domain and range of slots are defined in Table 4.1.

Table 4.1 Domain and Range of Slots

Property	**Domain**	**Range**
isTaughtby	Course	Faculty
teaches	Faculty	Course
isEnrolledIn	Student	Course
hasSupervisor	Student	Faculty
isStudentof	Student	Department

Table 4.1 depicts the domain and range of slots. For the object property *isTaughtby*, the domain is Course and range is Faculty i.e. *Course isTaughtby Faculty*, example Mathematics *isTaughtby* Nillima.

7. Create instances

The last step is creating instances of classes in the Class Hierarchy [75]. Instances refer to creating individuals for classes defined for ontology. The instances for the defined class in *University* Ontology have been shown below:

Table 4.2 Instances for *University* Ontology

Class	**Instances**
Course	Mathematics, Business, Communication, Semantic Web, English, Business Intelligence, Parallel and Distributed Computing
Student	Akriti, Manisha, Pulkit, Naina
Faculty	Nillima, Sujata, Deepali, Saira

Table 4.2 depicts the instances created for various classes in *University* ontology. For example, for class *Course* the instances defined are Mathematics, Business, Communication etc.

These seven basic steps explain the development of Ontology taking domain of *University* with very few concepts considered for explanation.

4.2. ONTOLOGY DEVELOPMENT FOR SEMENGINE IN PROTÉGÉ

4.2.1 Problem Scenario

When a user wants to search for information on the specified query "laptop price range 20000 to 30000", it includes the results for review and the particular brand rather than the intent of the user to find all the laptops that are in range between 20000 to 30000. This makes the user to filter out the required contents consuming a lot of effort and time. The specified query is depicted in Figure 4.2.

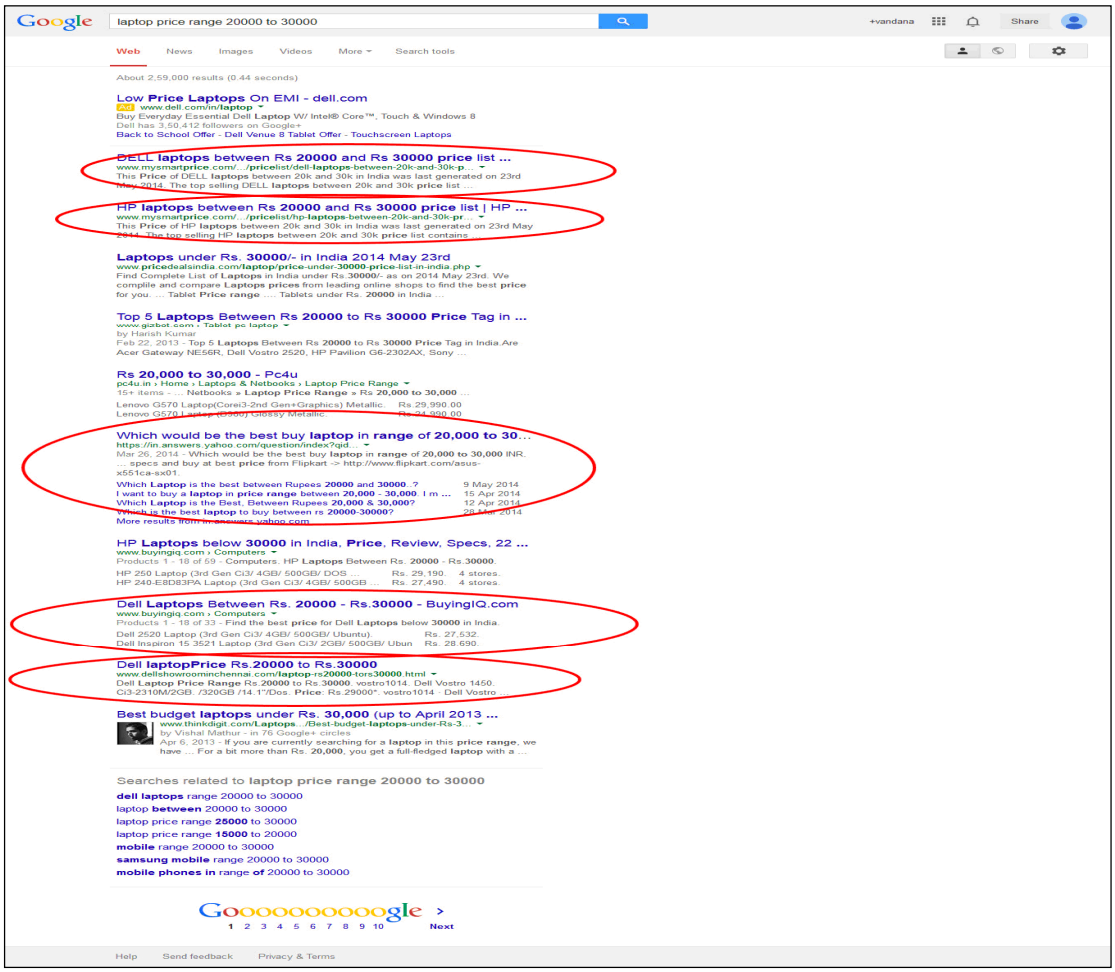

Figure 4.2 Google result for Query "laptop price range 20000 to 300000"

Figure 4.2 shows the Google results for the specified query "laptop price range 20000 to 30000", which includes the results for review and the particular brand rather than the intent of the user to find all the laptops that are in range between 20000 to 30000. The performance for the result is 50% for ten results. The result shows the sites giving imprecise results for a query.

Consider the performance for ten web pages for the described query. The analysis of results is as depicted in Table 4.3.

Table 4.3 Performance Rates for different Search Queries

Search Query	**Performance**
Laptop specification	60%
Laptop Purchase	80%
Laptop Ratings	30%
Laptop feedback	40%

It has been found that whatever query the user fire on laptop, most result sets contained the links to the selling sites of the laptop. Therefore laptop domain has been classified into three categories-

- Laptop Specification
- Laptop Selling
- Laptop Reviews

To avoid the above said problem, ontologies has been developed for these domains and use of these ontologies is extended for further annotation in research. This chapter deals with the development of ontologies in three categories and checking the consistency of ontology, and running DL (Description Logic) queries on the ontology to check the converge of ontology according to the set of query. For the research purpose, three ontologies have been designed in the domain of laptop to support the research work which is utilized for information extraction. For the development of ontology, an iterative development process as discussed earlier in the ontology engineering phase has been followed.

4.2.2 Ontology Development

The Ontologies developed for research purpose in the laptop domain are :

I. Ontology for Laptop_Review

II. Ontology for Laptop_Specification

III. Ontology for Laptop_Seller

I. Ontology for Laptop Review

This ontology deals with the reviews of laptop and has concepts related to review like rating, feedback and the various instances related to review sites are covered.

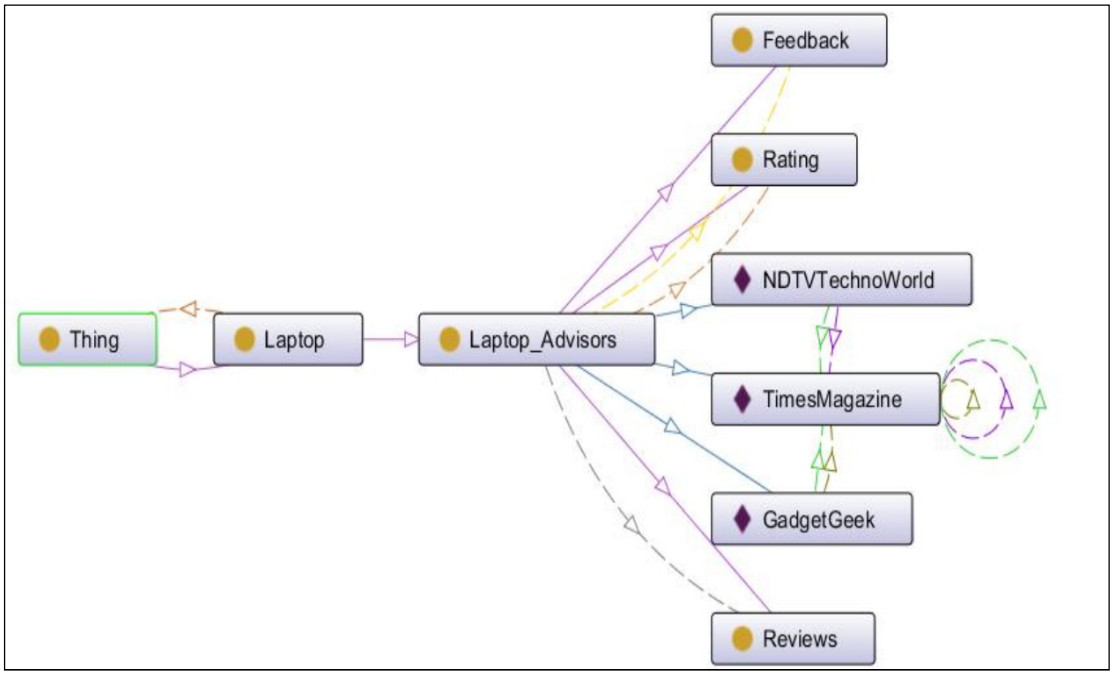

Figure 4.3 OntoVisualizer Result of Laptop_Review Ontology

Figure 4.3 shows the OntoGraph Vizualizer of *Laptop_Review Ontology*. In this Ontology the classes reviews, feedback and rating and the instances such as NDTV TechnoWorld, Times Magazine and GadgetGreek were developed which gives the reviews on the laptop. The different classes for the *Laptop_Review Ontology* are as depicted in Table 4.4.

Table 4.4 Classes for Laptop_Review Ontology

Class
Laptop_Advisors
Rating
Feedback
Reviews

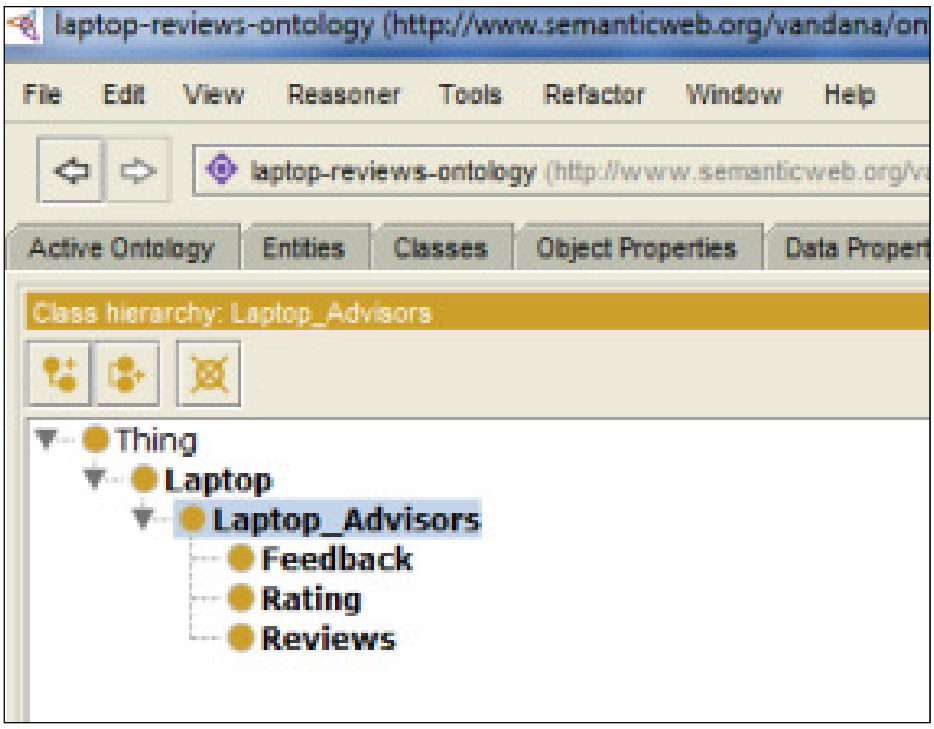

Figure 4.4 Laptop_Review Ontology Class.

Figure 4.4 depicts the classes of *Laptop_Review Ontology* created in Protégé. The instances of *laptop_Review* is depicted in Table 4.5

Table 4.5 Instances for Laptop_Review

Class	**Instances**
Laptop_Advisor	NDTVTechnoWorld
	TimesMagazine
	GadgetGeek

Table 4.5 shows the instances for the *Laptop_Advisor* class of *Laptop_Review Ontology* .

II. Laptop_Specification ontology

Ontology for *Laptop_Specification* developed in Protégé contains the concepts related to the specification of laptop for example laptop brand, laptop camera, laptop dimensions which are specified as classes. The different classes and subclasses for described for *Laptop_Specification Ontology* (see Table 4.6).

Table 4.6 Classes and SubClass of Laptop_Specification Ontology

Class	SubClass
Laptop_Audio	Microphone
	SteroSpeakers
Laptop_Brand	NIL
Laptop-Camera	NIL
Laptop_Dimensions	Laptop_Height Laptop_Width
Laptop_Display_Size	Display_13inch Display_14inch Display_17inch
Laptop_Memory	RAM ROM
RAM	RAM_4G RAM_8G
Laptop_OS	OS_Linux OS_Windows
Laptop_Processor	Processor_IntelCore_I3 Processor_IntelCore_I5 Processor_IntelCore_I7
Laptop_Wireless	Laptop_Bluetooth, Laptop_Wifi

The above table 4.6 specifies classes and subclasses of *Laptop_Specification* ontology and Table 4.7 depicts instances for ontology *Laptop_Specification* Ontology.

Table 4.7 Instances for Laptop_Specification Ontology

Class	Instance
Laptop_Brand	Dell, Compaq, HP, Sony, Apple
Display_13inch	Dell
Display_14inch	Compaq
Display_17inch	Sony, Apple
OS_Windows	Windows 98,Windows XP

Table 4.7 depicts for instances for different classes of *Laptop_Specification Ontology*. For example *OS_Windows* class has *Windows 98, Windows XP* as the instances.

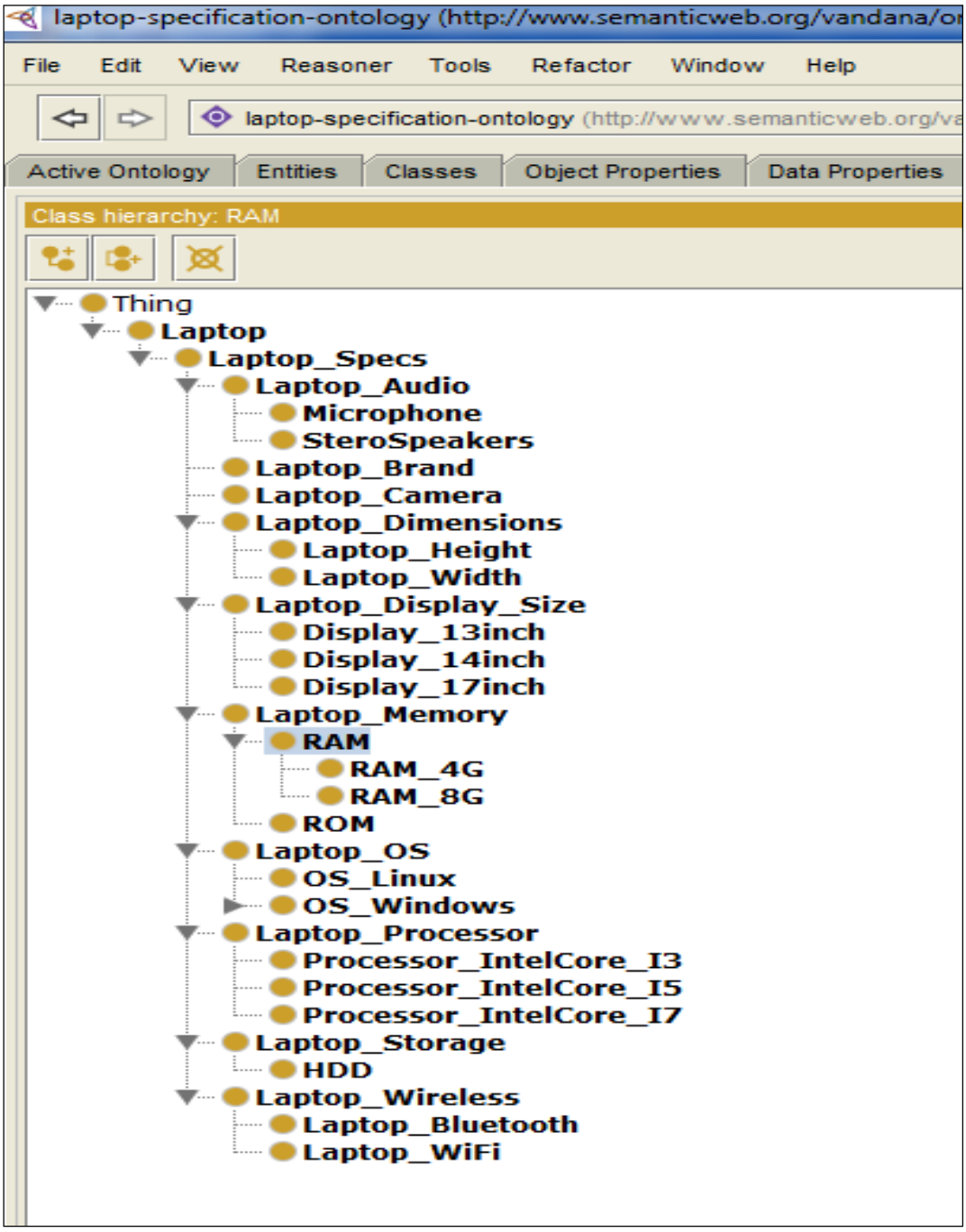

Figure 4.5 Laptop_Specification Ontology

Figure 4.5 shows the *Laptop_Specification Ontology* in which the concepts related to laptop like storage, dimensions, memory, wireless features, camera, brand, OS, display size are covered and the different object and data property relations between these classes and instances used were defined for these classes. The classes used in

this ontology are as depicted in Figure 4.5. The Ontovisualizer result of the ontology developed in Protégé is depicted in Figure 4.6.

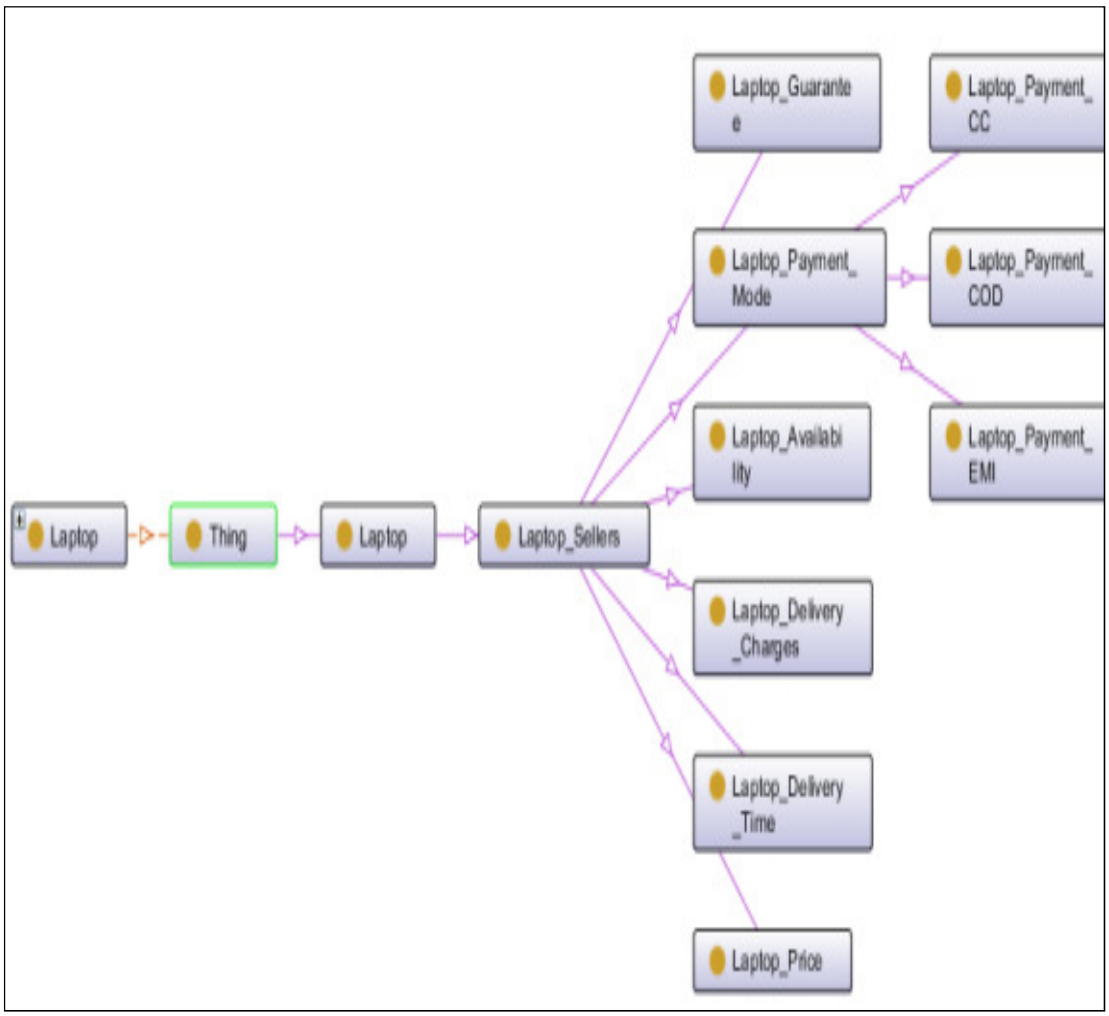

Figure 4.6 Ontovisualizer result of Laptop_Specification Ontology

III. Ontology for Laptop_Seller

The ontology for *Laptop_Seller* covers concepts related to the selling sites. It covers the concepts like laptop availability, laptop price, laptop delivery charges, laptop payment mode, and different laptop selling sites. The instances are defined for different laptop selling sites.

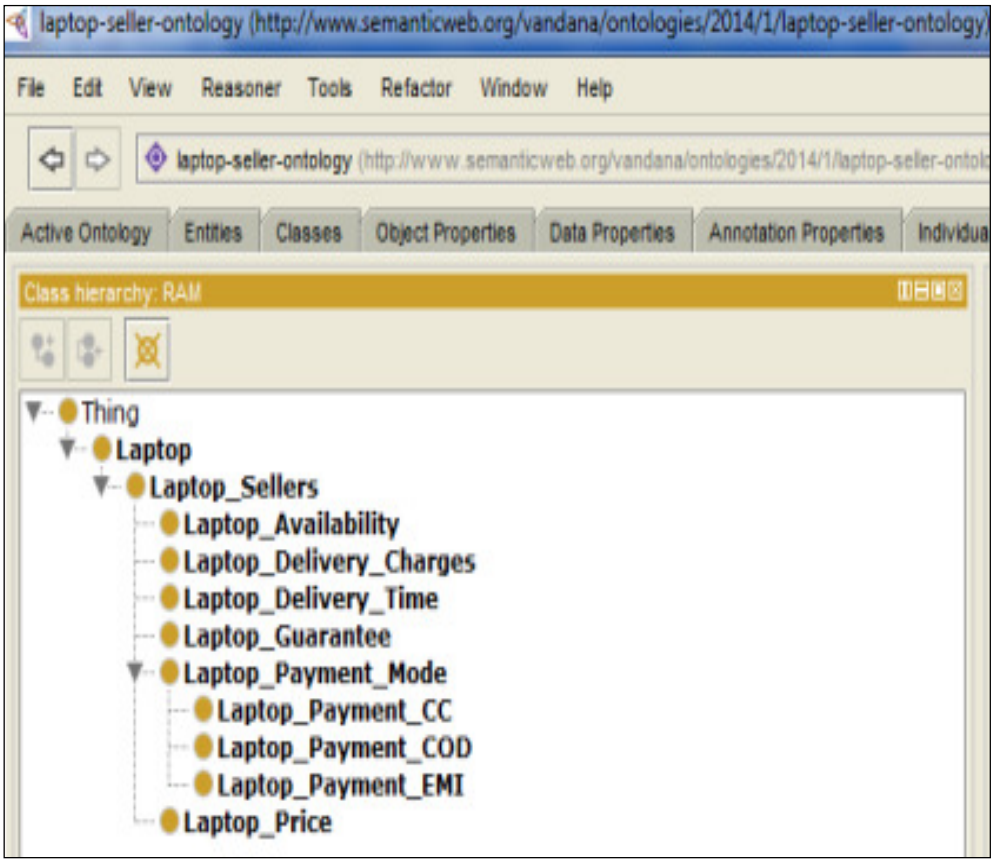

Figure 4.7 Laptop_Seller Ontology

Figure 4.7 Shows the development of *Laptop_Seller ontology* which covers the concepts related to selling of Ontologies and the individual instances covered in this ontology are Flipkart, ShopClues. The ontovisualizer result of *Laptop_Seller ontology* is depicted in Figure 4.8. Table 4.8 depicts the classes and subclasses of *Laptop_Seller* ontology .

Table 4.8 Classes and Subclasses of Laptop_Seller ontology

Class	**Subclass**
Laptop_Availability	NIL
Laptop_Deleviry_Charges	NIL
Laptop_deleviry_Time	NIL
Laptop_Guarantee	NIL
Laptop_Payment_Mode	Laptop_Payment-CC Laptop-Payment_COD Laptop_Payment_EMI
Laptop_Price	NIL

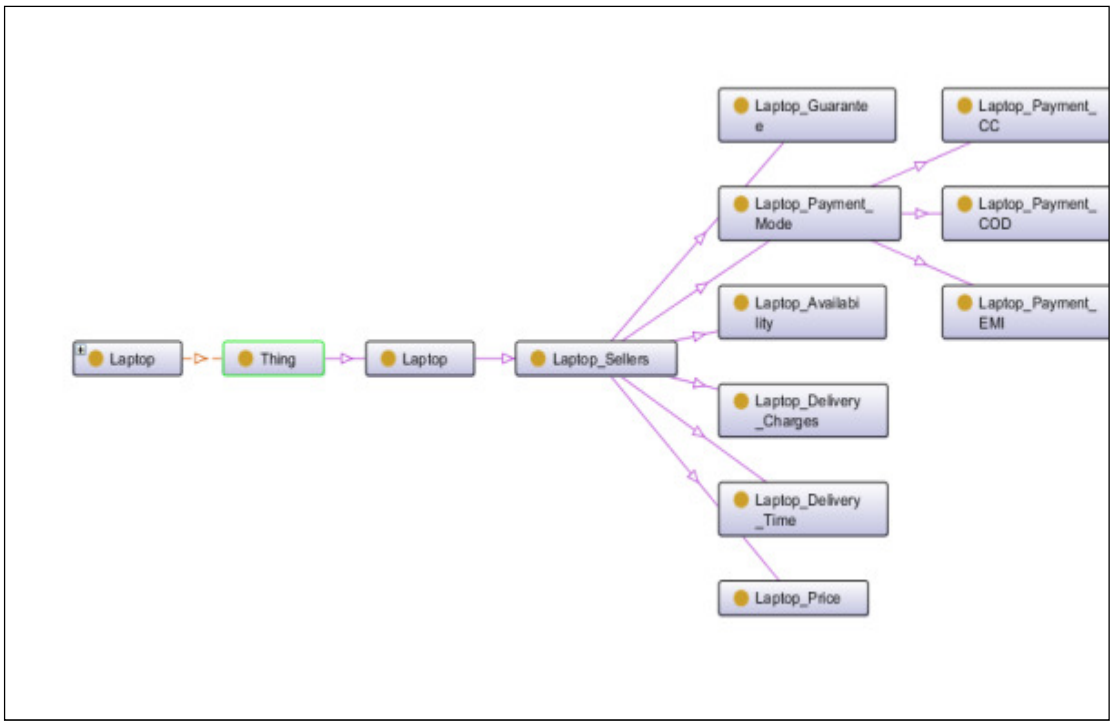

Figure 4.8 Ontovisualizer result of Laptop_Seller Ontology

Figure 4.8 shows the Ontovisualizer result of *Laptop_Seller Ontology* in Protégé. The developed ontologies are validated for correctness and can be queried using various query languages.

4.2.3 Query Retrieval in Ontology

In order to validate and verify the correctness of developed ontology, various query languages and rules languages can be used. The different query and rule languages for accessing ontology are

- DL Query [140]
- SPARQL Query [119]
- SWRL Rules[107]
- RDQL[121]

DL Query tab provides a powerful and easy-to-use feature for searching a classified ontology. It is a standard Protégé 4 plug-in, available both as a tab and also as a view widget that can be positioned into any other tab. The query language (class expression) supported by the plug-in is based on the Manchester OWL syntax[147], a user-friendly syntax for OWL-DL that is fundamentally based on collecting all

information about a particular class, property, or individual into a single construct, called a frame. DL Queries works only when the ontology is consistent, hence all the designed ontologies has been checked for consistency and found to be consistent using FACT++ reasoner [141].

In order to verify and validate the ontology in regard to different competency questions, Description Logic (DL) was used. The set of queries and DL Query format designed for the ontology developed in the domain of laptop has been indicated in Table 4.9.

Table 4.9 Set of queries to be executed in the domain of laptop using three ontologies.

Query	**DL Query**	**Ontology used**
Laptop with Display 13 Inches	Laptop and hasDisplay 13 inches	Laptop_Specification
Laptop with processor intel I3	Laptop and hasProcessor value " I3"^^string	Laptop_Specification
Storage capacity more than 1 TB	Laptop and hasstoragecapacity value "1TB"^^string	Laptop_Specification
Laptop with Price Range 30,000 to 40,000	Laptop_Price and hasrange30000 to 40000	Laptop_Seller
Costs less than 40,000	Laptop_Price and lessthan40000	Laptop_Seller
Payment Modes for Laptop Purchase	Laptop and haspaymentmode credit	Laptop_Seller
Laptop with Good rating	Laptop and hasrating " A"^^string	Laptop_Review

The query mentioned in Table 4.9 has been executed with the set of these questions on defined ontologies in the laptop domain. One of the results of DL query executed on developed ontology is depicted in Figure 4.9.

Figure 4.9 shows the result of execution of query for *Laptop_Specification Ontology*.

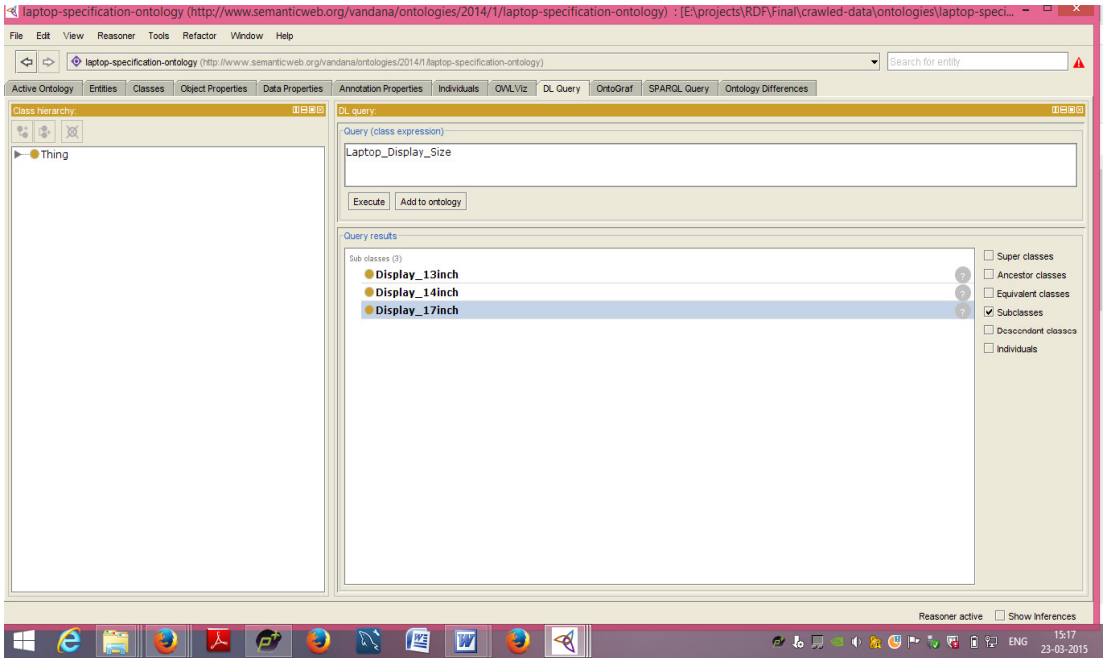

Figure 4.9 Execution of Query for Laptop_Specification Ontology

For the query "Laptop_Display_Size" depicts all the subclasses of the *Laptop-Display_Size* as shown in Figure 4.9. The correct result of DL query indicates that the ontology has been well designed and returns the results for individuals and classes.

Building domain ontologies is not a simple task as it requires a lot of effort and time to invest in domain conceptualization. Three ontologies were designed in the domain of laptop for ontology development and for checking the coverage of the concepts, set of queries were generated and implemented with the help of Description Logic. In next chapters, the developed ontology is used for annotation has been discussed in detail. The next chapter discusses about proposed SemCrawl framework.

CHAPTER V

5. SEMCRAWL: A FRAMEWORK FOR CRAWLING ONTOLOGY ANNOTATED WEB DOCUMENTS

5.1 INTRODUCTION

With the current feature of the web where everybody can publish its own content, there is large amount of data availability on the web, leading to unstructured information and a lot of data volume. This problem poses difficulties in relevant information retrieval, where large amount of information is presented to the user but user has to navigate through a large result set and find relevant information. Hence, there is a need for structured information on the web, which represent a well-defined meaning of the contents so that contents presented to the user have meaningful based matching of the query rather than results based on keyword matching of the query.

This problem has raised the need for semantic information representation over the WWW. There are different languages for semantic knowledge representation such as RDF, OWL. These languages have high expressive constructs and inference logic as compared to HTML which is just a presentation language.

The general purpose crawlers which are used for crawling web contents cannot be used for crawling Semantic Web contents for the following specified reasons:

1. In Semantic Web, there is a meaning based linkage of contents as compared to keyword based linkage in traditional web.
2. The language used to represent Semantic Web have different constructs as compared to traditional web.
3. Semantic Web is represented by heterogeneous set of resources-HTML annotated with RDF, OWL documents, DAML+OIL documents.

Thus, Semantic Web crawlers are mandate for crawling the semantic annotated documents.

The main characteristics features for Semantic Web for efficient retrieval are as follows:

1. The web should be represented with well-defined knowledge representation languages which have expressiveness power.
2. The web should be annotated with broad conceptual coverage of ontologies.
3. The web contents should be annotated with relevant ontologies, which will lead to high precision rate.

5.2 ANALYSIS ON CRAWLERS FOR SEMANTIC WEB

Slug crawler [148] is designed for harvesting Semantic Web contents. This crawler is implemented in Java using the Jena API. The features of this crawler is that it fetches the RDF files and follows *rdfs:seealso* links for further fetching of the contents. This crawler is designed as command based crawler system. The limitation of this crawler system is that it does not provide the methods for reusing of the crawled data and does not provide the evaluation details of the system.

Semantic Crawler based on CBR algorithm [149] harvest the metadata from the Semantic Web contents and clusters these metadata contents by annotating them with similar ontological concepts.

Swoogle crawler [111,112,113,150] harvest, parse and analyze Semantic Web contents which is annotated within web contents. It extracts metadata for the discovered document, computes relations and similarity between documents using N-grams or URIrefs as keywords.

Ontotext RDF crawler [151] provides a high level java programmable interface design which downloads the RDF contents from the WWW and builds a knowledge base. The implementation details of this includes the list of URIs, URI filtering parameters like depth of the pages to be crawled are maintained at every phase of crawler design.

RDF crawler [152] is a multithreaded implementation for downloading interconnected fragments of RDF data from the web creating a knowledge base and is capable of downloading from multiple sources. It takes care of deletion and replacement of RDF triples if it hit the same URL twice, keeping the data up to date every time.

The semantic crawler research is still in the beginning stages and currently not too many semantic crawlers have been developed. Most semantic crawlers do not provide their evaluation details [149, 153].

The above literature studied are particularly for crawlers that are designed for Semantic Web contents, all of these crawlers are able to extract Semantic Web contents but does not focuses on crawling the web contents which are annotated with knowledge representation techniques called ontology which refers to better representing a web page with ontological concepts and hence, helps in semantic retrieval of information.

This chapter deals with the development of architecture that crawl the ontology annotated documents, filtering out the semantically annotated web pages and extracting the triples relations from the underlying annotated ontology associated with the web page.

5.3 OBJECTIVE OF PROPOSED FRAMEWORK OF SEMCRAWL

The proposed SemCrawl framework has the following objectives:

1. Crawling heterogeneous documents

Crawling heterogeneous documents from the web which are specified in semantic representation language such as RDF, OWL, RDF embedded with HTML.

2. Filtering the contents

Only the web contents that have semantic knowledge representation needs to be parsed and other contents need to be filtered out.

3. Parsing the ontology annotated documents

Parsing of the ontology annotated documents into triple relations <S, P, O> where S stands for Subject, P stands for Predicate and O stands for Object which allows the logical inferences to be made in future.

5.4 PROPOSED ARCHITECTURE OF SEMCRAWL

SemCrawl [48] downloads the web pages which are annotated with ontologies, and filters the pages which are not annotated with ontology. The proposed architecture of SemCrawl in Figure 5.1 consists of the following functional modules:

a. Fetch module

b. Filter module

c. Link extraction Module

d. URI Dispatcher Module

e. Parser Module

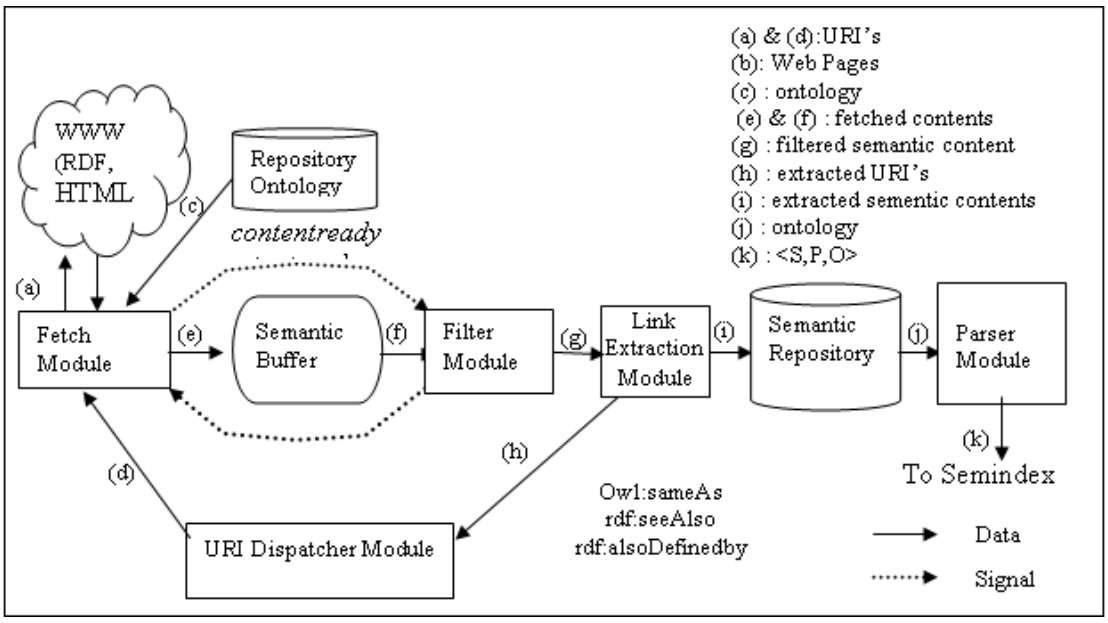

Figure 5.1 Architecture of Proposed SemCrawl

A. FETCH MODULE

This module fetches HTML, RDF, OWL web pages and their corresponding ontologies associated with the web. After fetching the contents, these contents are stored in Semantic Buffer. When the contents are available in Semantic Buffer, *contentready* signal is sent to Filter Module which then filters the web pages having no semantic annotation.

The pseudo code of Fetch Module has been shown in Figure 5.2.

```
Algorithm Fetch Module(input: URIs, output: fetched contents)
{
    wait (contentempty);
    extract URI from the URI Queue ;
    fetch the contents from web;
    store the contents in semantic buffer;
    signal (contentready);
}
```

Figure 5.2 Pseudo code for Fetch Module

Fetch Module waits for *contentempty* signal from the Filter Module. After receiving the *contentempty* signal from the Filter Module, it fetches the contents from the web, based on URIs received from URI Dispatcher Module. When the contents are fetched from the web, the Fetch Module sends these contents to the Semantic Buffer signalling *contentready* signal to Filter Module for further processing.

B. FILTER MODULE

This module waits for the *contentready* signal from Fetch Module. On receiving the *contentready* signal from the Fetch Module, this module filters the contents having no annotation and the filtered contents are sent to the Link Extraction Module for further extraction of the links from the downloaded web pages.

```
Algorithm Filter Module (input: semantic buffer, output: filtered pages)
    {
        Wait (contentready);
        get  the contents from buffer;
        extract the semantic  web pages;
        sends the filtered contents to link extraction module;
        signal (contentempty);
    }
```

Figure 5.3 Pseudo code for Filter Module

The pseudo code for Filter Module has been shown in Figure 5.3. The Filter Module waits for the *contentready* signal from the Fetch Module. After receiving the signal

from the Fetch Module, it get the contents of Semantic Buffer as input, extracts the Semantic Web pages and input the filtered contents to Link Extraction Module for further processing. After the specified work, Filter Module signals *contentempty* signal to the Fetch Module for further fetching of contents from the web.

C. LINK EXTRACTION MODULE

Upon receiving the contents from the Filter Module, Link Extraction Module extracts the links from the contents and sends it to URI Dispatcher Module, which stores all the links in queue and fetch the corresponding contents from those links. The pseudo code for Link Extraction Module has been depicted in Figure 5.4.

```
Algorithm Link Extraction Module (input: filtered contents, output: extracted
                                                        links and contents)
{
    if (Owl: sameAs or rdf: seeAlso or rdf: alsoDefinedby constructs in contents)
     send those links to URI Queue for further processing
    else
     store the web page and ontology in semantic repository database for further
            extraction of concepts;
}
```

Figure 5.4 Pseudo code for Link Extraction Module

The Link Extraction Module extracts the links from the contents and if the constructs are *Owl: sameAs* or *rdf: seeAlso* or *rdf: alsoDefinedby* are found in the contents, those URLs links are sent to URI Dispatcher Module for further fetching of contents from the web.

D. URI DISPATCHER MODULE

This module gets the URIs from the Link Extraction Module. These URIs links are stored by URI Dispatcher in a queue data structure and follows the First-in-First-out policy. The contents are fetched for a specified URL and are given to Fetch Module for further storage into Semantic Buffer. Example for a URI Dispatcher module is as shown in Figure 5.5.

Figure 5.5 Structure of a URI Dispatcher Module

The structure of URI Dispatcher Module has been depicted in Figure 5.5 which is queue data structure and consists of a set of URLs which are given as input to the Fetch Module for fetching of contents.

E. PARSER MODULE

This module gets the input from the Semantic Repository and parses the ontologies into <S, P, O> format where S refers to Subject, P refers to Predicate and O refers to Object.

```
Algorithm Parser Module (input: semantic buffer, output: parsed output)
{
extract the crawled ontology  from semantic repository;
extract subject, predicate and object;
store triples in database with three columns subject, predicate, object
      <S, P,O>;
 }
```

Figure 5.6 Pseudo code for Parser Module

The pseudo code for Parser Module has been depicted in Figure 5.6. The crawled data from the Semantic Repository has been provided as input to the Parser Module which extracts Subject, Predicate and Object called as triple relation, parsed using Jena APIs. These triples are stored in MYSQL database as <S, P, O>, as depicted in Figure 5.7.

Subject *: http://www.semanticweb.org/vandana/ontologies/2014/1/laptop-reviews-ontology#Laptop_Feedback*

Predicate*: http://www.w3.org/2000/01/rdf-schema#subClassOf*

Object *: http://www.semanticweb.org/vandana/ontologies/2014/1/laptop-reviews-ontology#Laptop_Advisors*

Figure 5.7 <S, P, O> Triples

Figure 5.7 shows <S, P, O> triples of *Laptop_Feedback* ontology parsed into Subject, Predicate and Object which shows *Laptop_Feedback* is subclass of *Laptop_Advisors.*

5.5 METHODOLOGY FOR THE DEVELOPMENT OF SEMCRAWL ARCHITECTURE

The methodology followed for the development of SemCrawl Framework is depicted in Figure 5.8

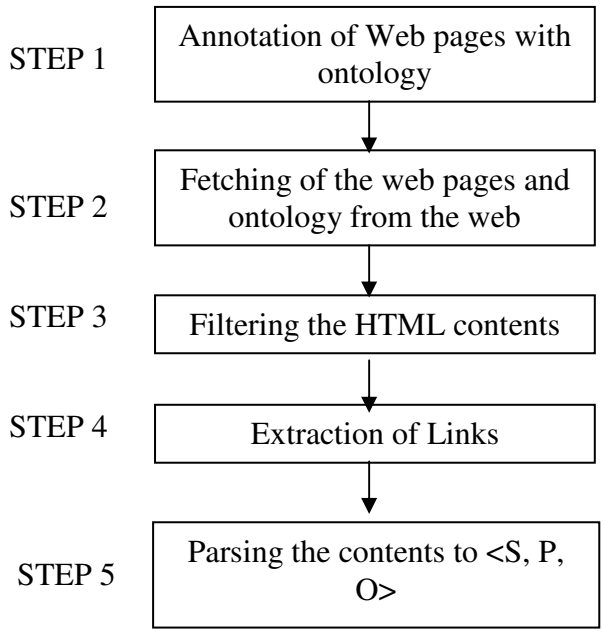

Figure 5.8 Methodology for development of SemCrawl Architecture

The following are the steps for methodology followed for development of SemCrawl:

STEP 1. Annotation of Web pages with ontology

The web contents are annotated with ontology. Three ontologies have been designed in laptop domain- *Laptop_Seller* ontology, *Laptop_Review* ontology, *Laptop_*

Specification ontology using Protégé framework. These ontologies are annotated with the web pages related to them. For example *Laptop_Specification* ontology is annotated with [73] using link href= “ ” attribute with webs page *http://www.pcworld.com/article/187749/laptop_buying_guide_making_sense_of_the_specs.htm.*

STEP 2. Crawling of the web contents of different sources

Upon receiving the URI from the URI from URI Dispatcher, the contents are crawled from the web which is HTML web pages, Semantic Web pages represented in Semantic Web languages such as RDF, OWL, which is then stored in Semantic Buffer and is given as input for filtering, link extraction and parsing.

STEP 3. Filtering the HTML contents

The HTML contents are filtered from the repositories, only the web pages which are annotated with ontology are sent for further extraction of links and the contents that are filtered contents have been stored to Semantic Repository.

STEP 4. Extraction of Links

The links are extracted from the web contents and are given as input for further fetching. The *rdf:seealso*, *Owl:sameAs* construct are used for extraction of links which is then given to URI Dispatcher module for further fetching of contents.

STEP 5. Parsing the contents to <S, P, O>

The contents are parsed to <S, P, O> where S refers to subject, P refers to predicate and O refers to object. This parsing of the contents represent that every RDF statement has a well-defined meaning and is semantically related to each other which helps in finding out the inferences.

5.6 EXPERIMENTS AND RESULTS

SemCrawl was implemented in Java using Jena APIs for parsing and Eclipse framework for project development. For the experiment purpose 100 pages were downloaded, out of which 20 web pages were not relevant and 80 pages were relevant to the scope of the work and these pages were annotated with relevant ontologies. The

ontology annotation helps in finding relation between entities of the web page which helps in producing relevant results.

The repositories of web pages shown in Figure 5.9 has been crawled.

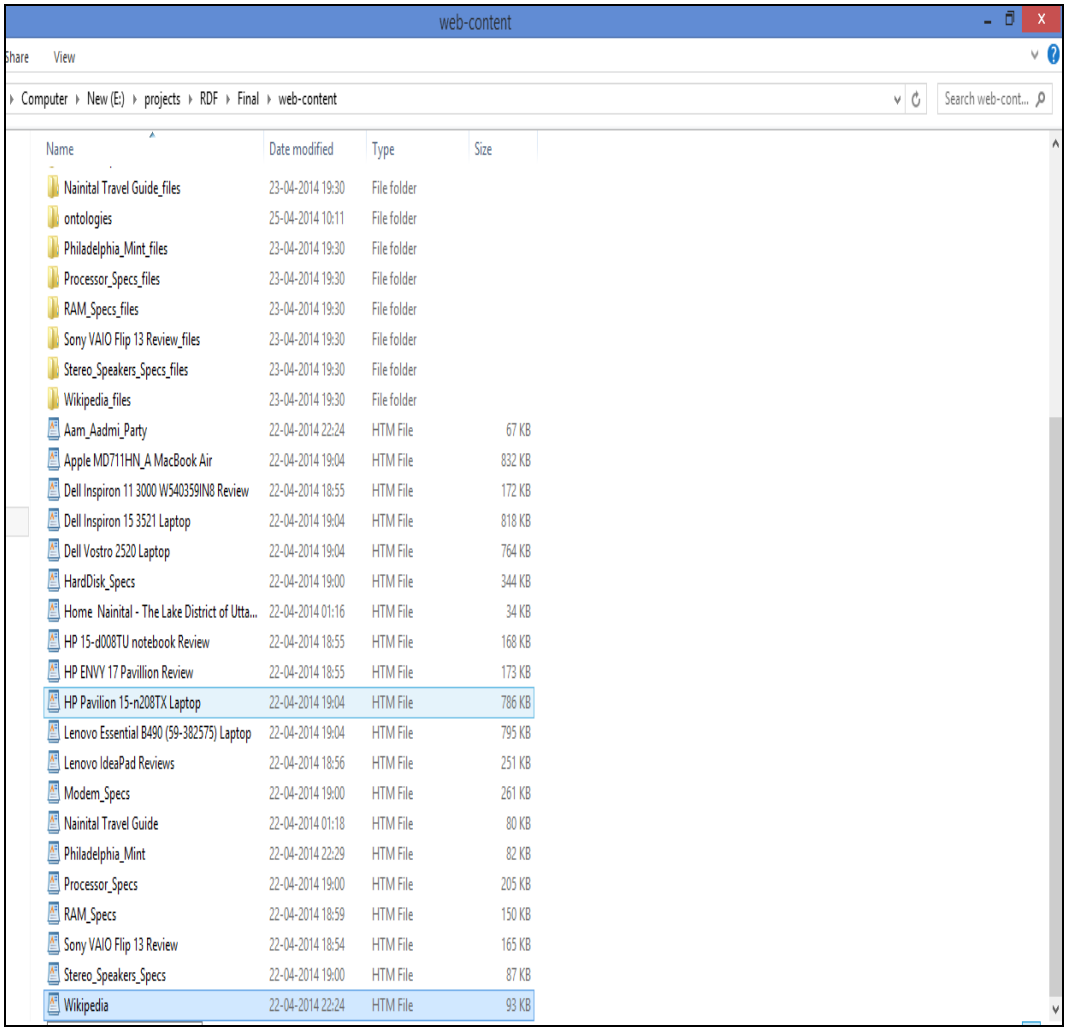

Figure 5.9 Repositories of web pages

The repositories consisting of web pages annotated with ontology and simple HTML pages were crawled. These web pages are provided to Filter Module that filters out the pages having no semantic annotation to it.

The repository shown in Figure 5.9 was given as input to Filter Module that filters out the pages with no annotation to it and has a set of web pages which are annotated with ontology. The output of Filter Module consists of all the web pages that are associated with ontology as shown in Figure 5.10.

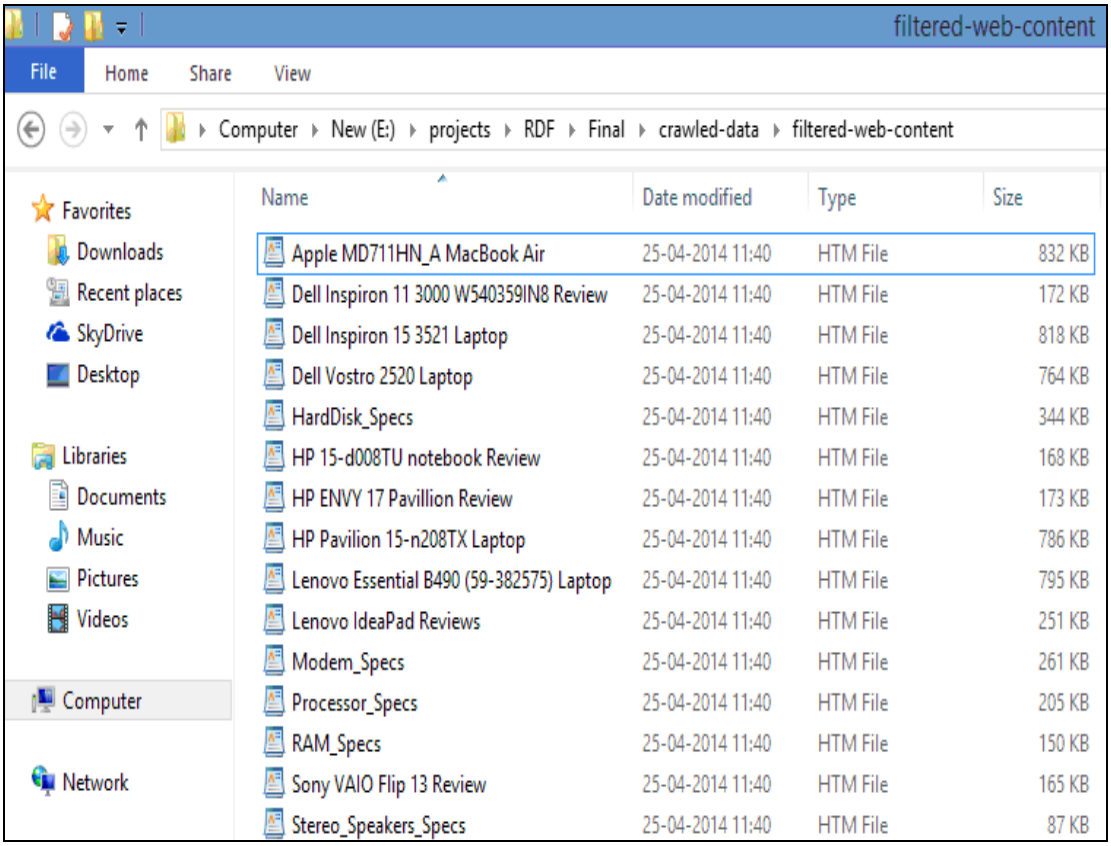

Figure 5.10 Filtered web pages which are annotated with ontology.

The implemented proposed system for SemCrawl framework crawled all the RDF pages marked-up with ontology and filter out HTML pages which were without annotation. SemCrawl has been implemented with the series of three tests Test1, Test2 and Test3 with each test taking a collection of 20 pages, 50 Pages and 100 Pages respectively, with each test 100% accuracy was achieved. The SemCrawl is able to crawl all the relevant contents from the web discarding the unwanted HTML web pages. Table 5.1 indicates the results of the various tests conducted on SemCrawl.

Table 5.1 Series of Test conducted on Repositories for Crawler Module

Test	Repositories	Type of Repositories	Crawler Output
Test 1	20 Pages	RDF Web Pages	20 Pages
Test 2	50 Pages	RDF,OWL,HTML Pages	38 Pages with filtered out 12 HTML Pages
Test 3	100 Pages	RDF,OWL,HTML Pages	85 Pages with filtered out 15 HTML Pages

The Parser Module was implemented in Java using Eclipse framework and Jena API library. Jena is convenient toolkit to manipulate RDF models for developing application within Semantic Web [154]. Parser module extracts triples from the ontology. Figure 5.11 depicts the screenshot for extracting triples from the ontology using Jena Library in Eclipse Development Framework.

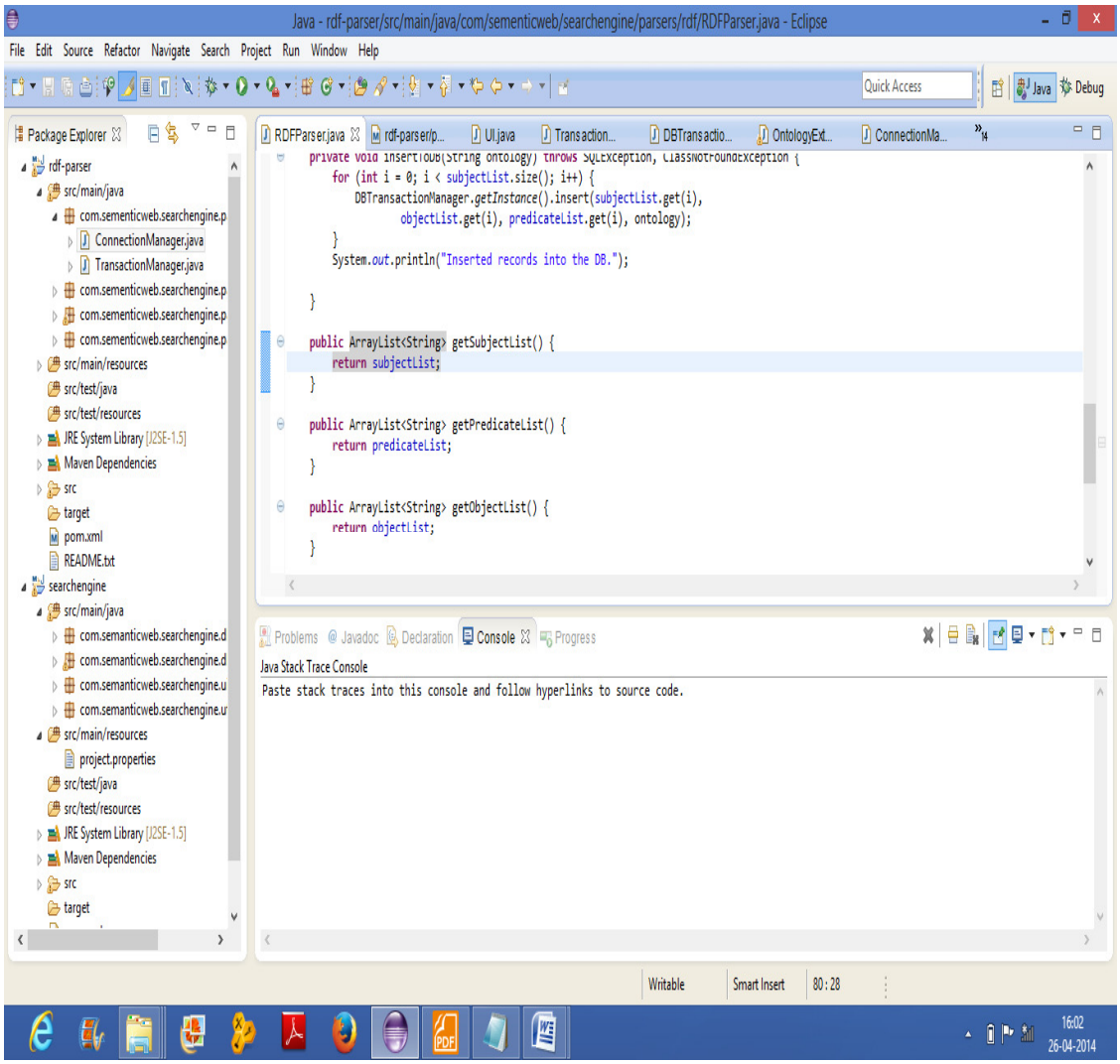

Figure 5.11 Extraction of Subject, Predicate, Object from Ontology

Triples are extracted in the form <Subject, Predicate, Object> as depicted in Figure 5.11. The total number of triples depends on the vocabulary of designed ontology. Figure 5.12 shows the output of Parser Module showing the triples relation<S, P, O>.

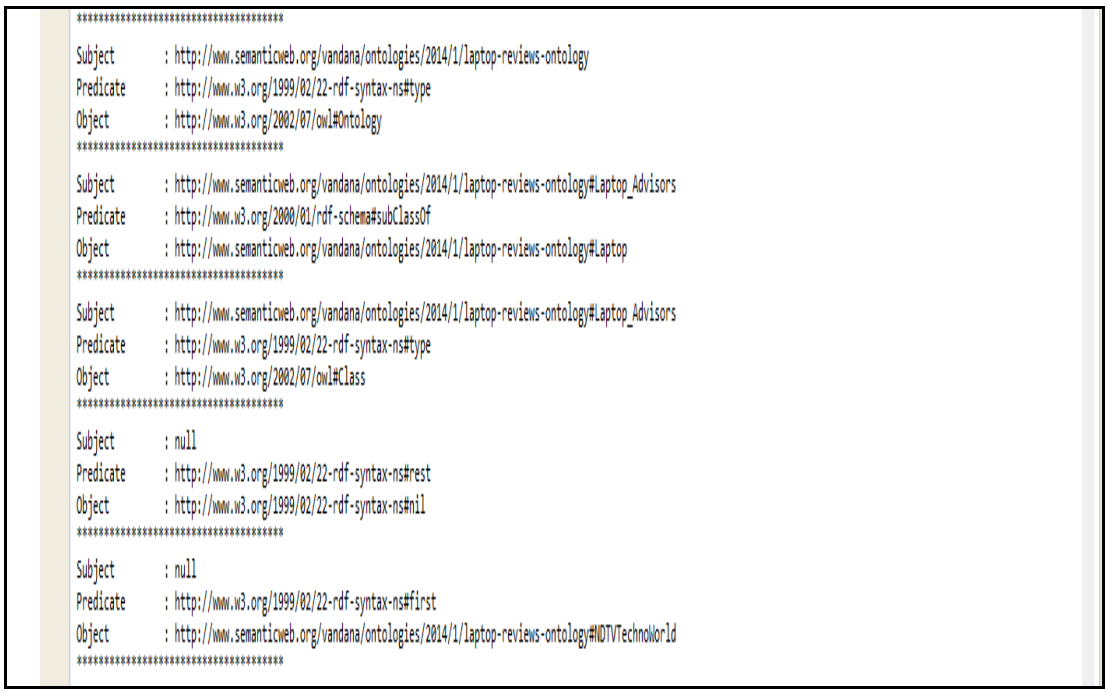

Figure 5.12 Console output of Extracted Subject, Predicate, Object <S, P, O> Triples

Figure 5.12 depicts ontologies were parsed into <S, P, O> specifying a triple relation for the specified ontological concepts. The implemented system was analyzed and 1546 number of triples was discovered.

The SemCrawl Module successfully fetched the web pages and filters the Semantic Web contents for further processing, extracted the link for extraction of web pages from the web, and parses the contents into triples. This module has the feature of extracting heterogeneous documents from the web which can be further indexed and therefore helps in the semantic retrieval of the information. The crawled repository was given as input for further indexing for efficient information retrieval.

The next chapter discusses about the indexing architecture SemIndex.

CHAPTER VI

6. SEMINDEX: AN EFFICIENT INDEXING MECHANISM FOR ONTOLOGIES

6.1 INTRODUCTION

The ontologies in the Semantic Web are generally represented in RDF, OWL like language that has high expressiveness and inference constructs. Ontologies expressed in these languages can be represented in following three ways in Semantic Web-

- RDF/XML Code
- Triples
- Graph

6.1.1 RDF/XML Code The RDF statements are represented by RDF /XML code. Consider RDF/XML document represented in Figure 6.1.

```
</xml version="1.0" encoding="UTF-16"?>
<rdf: RDF
xmlns: rdf= "http://www.w3.org/1999/02/22-rdf-syntax-ns#
xmlns:rdfs= "http://www.w3.org/2000/01/rdf-schema#>
<rdf: Description rdf: about= "CIM2"
<coursename>Mathematics for Technology</coursename>
<istaughtby rdf: resource = "http://university.org/faculty-detail-drarvind"/>
</rdf: Description>
</rdf: RDF>
```

Figure 6.1 RDF/XML Code

Figure 6.1 shows an example of simple RDF/XML document representing a RDF statement "**CIM2 is Mathematics for Technology and is taught by Dr. Arvind**". The various language constructs used in this example has been discussed as below-

- *rdf: Description* indicates the beginning of RDF statement.

- *rdf: about* indicates the Subject of RDF statement(*rdf: about="CIM2"*).
- tag represents the Predicate of RDF statement(<coursename>,<istaughtby>).
- *rdf: resource* here indicates the Object of RDF statement (*rdf:resource=http://university.org/faculty-detail-drarvind).*

6.2.2 Triples- RDF is a logical data model for representing set of specific resources each provided with a pair of properties and property values. The data model for RDF language representation has following three main components:

(i). **Resource** is anything that can have a URI; this includes all the web pages, as well as individual elements of an XML document.

(ii). **Property** is a resource that has a name and can be used as a property.

(iii). **Statement** is the combination of a resource, a property, and a value. These are also known as the 'subject', 'predicate' and 'object' of a statement and are represented as <S, P, O>.

Triples <S, P, O> represent the relations between the Subject and Object through Predicate. For the example discussed in Figure 6.1 the triple <S, P, O> has been shown in Figure 6.2.

***Subject**: CIM2*
***Predicate**: istaughtby*
***Object**: http://university.org/faculty-detail-drarvind*

Figure 6.2 Triple representation of RDF/XML Document

The example shows the triple representation of an RDF/XML document.

6.2.3 Graph- Various set of statements can be represented as labelled directed graph, with nodes of the graph representing as Subject and Object and arc represented as Predicates. Each part of the node can be a URI, string element. The graph representation for the RDF/XML code example depicted in Figure 6.1 has been shown in Figure 6.3.

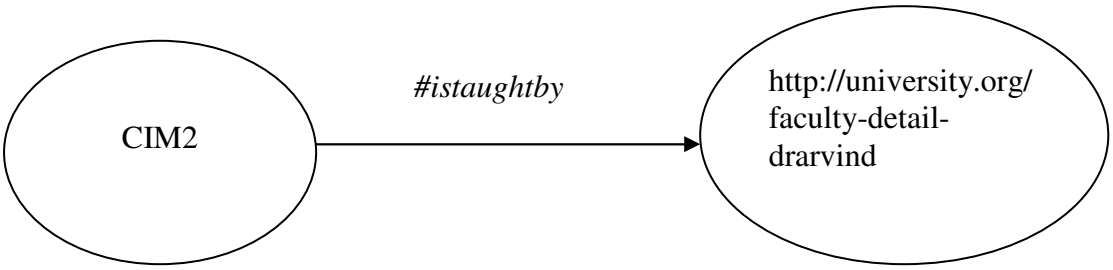

Figure 6.3 Graph Representation for RDF/XML document

The graph indicates CIM2 is subject; *(http://university.org/faculty-detail-drarvind)*is the object; connected by the predicate *#istaughtby.*

The approach for retrieval of web contents represented in HTML cannot be directly applied to web contents associated with ontologies because of different approach of representation and interpretation. Due to this reason, traditional approaches for indexing like N-gram indexes, positional indexes cannot be applied to knowledge represented in RDF, OWL. Thus, there is need for different indexing mechanism, dealing with the knowledge representation languages like RDF, OWL. Mature relational database is regarded as a good basis for RDF store to manage large scale of triples.

This chapter deals with comparative analysis of existing RDF Indexing techniques based on various parameters, proposed architecture for indexing mechanism, describing the experimental results of the proposed method.

6.2 COMPARATIVE ANALYSIS OF INDEXING TECHNIQUES

RDF indexing structures is based on three categories : Triple Store (TS), Vertical Partitioning (VP) and Property Table (PT) [4]. In these indexing strategies, different techniques have been used for indexing. A comparative analysis of these three indexing strategies based on various parameters is described in Table 6.1.

Table 6.1 Comparative analysis of indexing Techniques

Strategy	Description	Space	Limitations of the scheme	Overhead involved	Implemented by systems
Triple Store[156]	In this approach RDF statement of the form (Subject, Property, Object) is stored in triples form; all triples are stored in a single table with three columns (S, P, O).	Space requirement is determined by the number of triples in the table [155].	1. Slow execution of queries as triples are stored in one single RDF table. 2. Requires many self joins because of the single large table.	The overhead is incurred in schema storage of the triple store table.	Used by systems like Redland [157], 3 store [158], RDFStore [159].
Vertical Partioning [160]	In this approach a different table is created for each distinct predicate to store all triples which represent that predicate.	Space requirement is determined by number of tuples in the tables and number of tables, which is the equal to the number of unique predicates in the RDF data [155].	Most of queries require joins or unions to combine data from several tables.	The overhead is incurred in storing the schema of each vertical partioning table.	Used by systems like SW Store [161].
Property Table [11]	In this approach data is stored in relational tables; it contains cluster triples containing properties defined over similar subjects.	Space requirement for table is more in comparison of triple table, because adding properties require adding more tables.	1. Queries require multiple union clauses and joins to combine data from several tables. 2. Inability to handle multi-valued attributes.	Reduces number of self joins.	Used by systems like Jena [163], RDFSuite [164], Sesame [165] and 4 Store [166].

Table 6.1 shows comparative analysis of the various indexing mechanism. Triple Store indexing mechanism is widely used as it is supported by many framework for storage. Therefore, the proposed architecture has been designed using Triple Store as

the base architecture for indexing which is discussed in the architectural model of SemIndex.

6.3 OBJECTIVES OF THE PROPOSED WORK

The proposed framework for indexing has the following objectives:

1. To create an index for RDF statements which helps in inference logic.
2. To create an index for HTML file with respect to Ontology.
3. The index created for the ontology annotated documents should be of smaller size in comparison to the index for the traditional web.
4. The index created should help in effective information retrieval.

6.4 PROPOSED ARCHITECTURE FOR INDEXING: SEMINDEX

The proposed architecture of SemIndex indexes the web pages that are annotated with ontologies. The proposed framework of SemIndex is depicted in Figure 6.4 and consists of the following main modules:

(i) SemCrawl Module
(ii) Indexer Module
(iii) Ontology Extractor Module
(iv) Quad Store RDF Index
(v) Onto_HTML Mapping Database

The output of SemCrawl Module is given as input to Ontology Extractor Module and Indexer Module. Ontology Extractor Module extracts HTML File name and its corresponding ontology which is stored in Onto_HTML Mapping Database as <HTML File, Ontology>. The Indexer Module extracts Subject, Predicate, Object of the corresponding ontology which is stored in Quad Store RDF index as <Subject, Predicate, Object, Ontology>. The result is returned for the query entered based on the searching of the specified indexes.

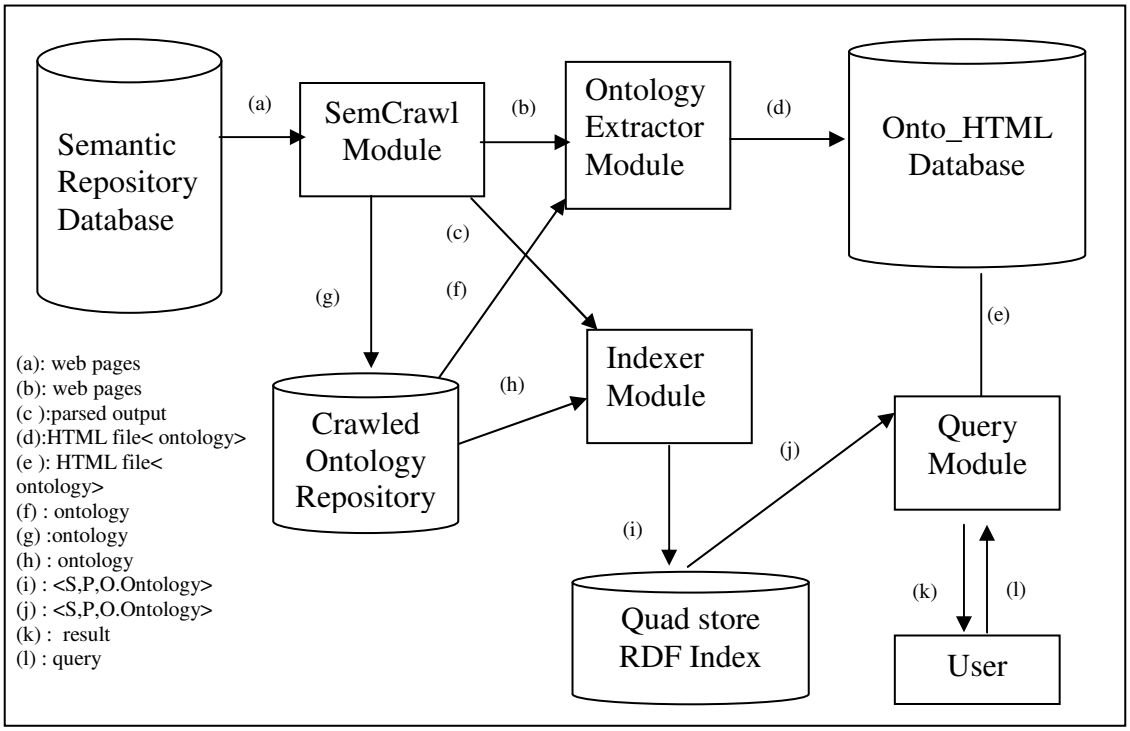

Figure 6.4 Architecture of SemIndex for indexing crawled Ontologies

The detailed functionality of functional modules used in SemIndex has been discussed below:

6.4.1 SemCrawl Module

As discussed earlier in Chapter 4, SemCrawl crawls the HTML web pages, web pages associated with ontology and domain ontology that is used to describe a web page. The crawler module also filters out the web pages which are just HTML pages whereas the pages which have ontology associated with them is stored in repositories. The output of SemCrawl Module acts as input to the SemIndex.

6.4.2 Indexer Module

The input of this module is the parsed output and the crawled domain ontology. The Indexer Module parses the associated ontology into <S, P, O, Ontology> which signifies Subject, Predicate, Object referring to Ontology, for which triples has been parsed. The RDF contents are parsed into Subject, Predicate, Object using Jena Parser [154].Figure 6.5 specifies the pseudo code for Indexer Module.

```
Algorithm Indexer Module(input: parsed output, output: index)
 {
read ontology file from the crawled  ontology  repository;
parse ontology file ;
extract RDF triples;
insert subject, predicate, object ,ontology value to database
}
```

Figure 6.5 Pseudo code for Indexer Module

The Indexer Module creates an index of <S, P, O, Ontology > after parsing the ontology file from the Crawled Ontology Repository of SemCrawl Module and triples.

Example of Indexer module

Consider the example of *Laptop_Seller* Ontology in Figure 6.6.

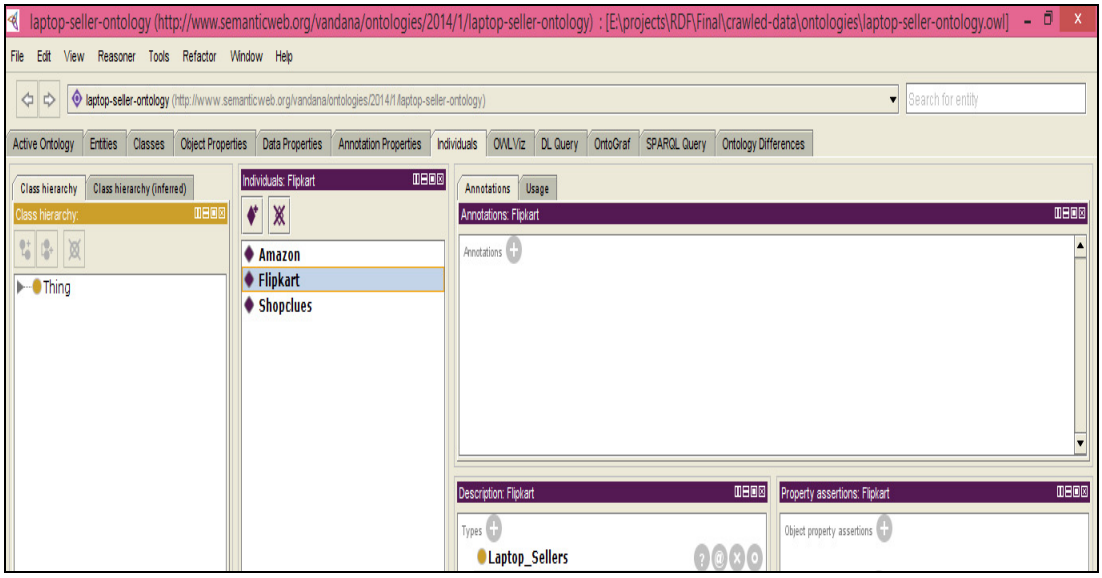

Figure 6.6 Instances of Laptop_Seller Ontology

For ontology *Laptop_Seller* , Figure 6.6 indicates having RDF statement “Flipkart is a type of Laptop_Seller”. This RDF statement is parsed into Subject, Object, Predicate as given in Figure 6.7.

Subject *: http://www.semanticweb.org/vandana/ontologies/2014/1/laptop- seller-ontology#Flipkart*

Predicate*: http://www.w3.org/1999/02/22-rdf-syntax-ns#type-of*

Object *: http://www.semanticweb.org/vandana/ontologies/2014/1/laptop-seller*

Figure 6.7 Example of a Triple <Subject, Predicate, Object>

Figure 6.7 shows the triple relation of <S, P, O> for the ontology *Laptop_Seller* where S is *Flipkart*, P is *type_of* and O is the *Laptop-Seller.* Graphical representation for the triple relation is as depicted in Figure 6.8.

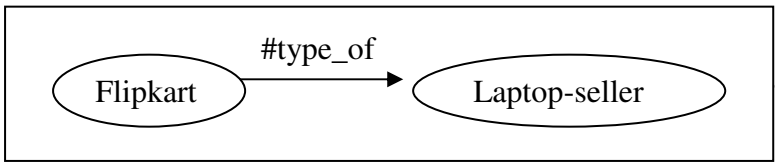

Figure 6.8 Triple representation

The example in Figure 6.8 specifies the triple relation between the *Laptop_Seller* and *Flipkart*, i.e. *Flipkart is a type_of Laptop-Seller.*

6.4.3 Ontology Extractor Module

This module extracts the ontology name associated to a particular ontology and indexes as <HTML File, Ontology> onto the database. Figure 6.9. Specifies the algorithm for ontology extractor module.

```
Algorithm Ontology Extractor (input: web pages, output: index)
 {
parse web pages from Semantic Repository;
parse ontology from crawled ontology repository;
extract the ontology name and associated HTML web page;
index (HTML File, Ontology) in the database;
}
```

Figure 6.9 Algorithm for Ontology extractor

Ontology Extractor Module parses the web pages and ontology, and extracts the HTML File name of that web page and the ontology associated web page. The created index < HTML file, Ontology> will be stored in database.

Example of Ontology extractor Module

HTML File	*: E:\projects\RDF\Final\crawled-data\filtered-web-content\amazon.htm*
Ontology	*: laptop-seller-ontology*

Figure 6.10 Example <HTML File, Ontology> index

Figure 6.10 depicts the example of <HTML File, Ontology> index, which shows that HTML web page, is associated with *Laptop_Seller* ontology.

6.4.4 Quad Store RDF Index

Quad Store consists of four indexes in the database Subject, Predicate, Object and the annotated ontology.

Subject	**Predicate**	**Object**	**Ontology**
Amazon	Type-of	Laptop_Seller	Laptop_seller Ontology

Figure 6.11 Index of<Subject, Predicate, Object, Ontology>

Figure 6.11 represents the index, where Predicate is *type_of* and associated ontology is *Laptop_Seller* which means the statement "Amazon is a type_of Laptop_Seller" is specified in *Laptop_Seller* ontology.

6.4.5 Onto_HTML Mapping Database

The Ontology Extractor Module extracts the ontology and the web page associated with it that is indexed in database as<HTML File, Ontology >.

HTML File	Ontology
Flipkart.htm	Laptop_Seller ontology
Stereo_Speakers_Specs.htm	Laptop_Specification-ontology

Figure 6.12 Index of < HTML File,Ontology>

Figure 6.12 represents ontology and HTML file having ontology associated with it. For example two entries are shown which shows that HTML file *Flipkart* has *Laptop_Seller* ontology associated with it. And HTML file describing stereo speaker specification has *Laptop_ specification* ontology associated with it.

6.5 EXPERIMENTS AND RESULTS

The SemIndex architecture was developed in Java using Eclipse framework and Jena APIs. MYSQL server was used for indexing. Taking input as the Semantic Repository Database, the Ontology is parsed to <S, P, O>. Figure 6.13 depicts the results, in which the ontology has been parsed into <Subject, Predicate, Object>.

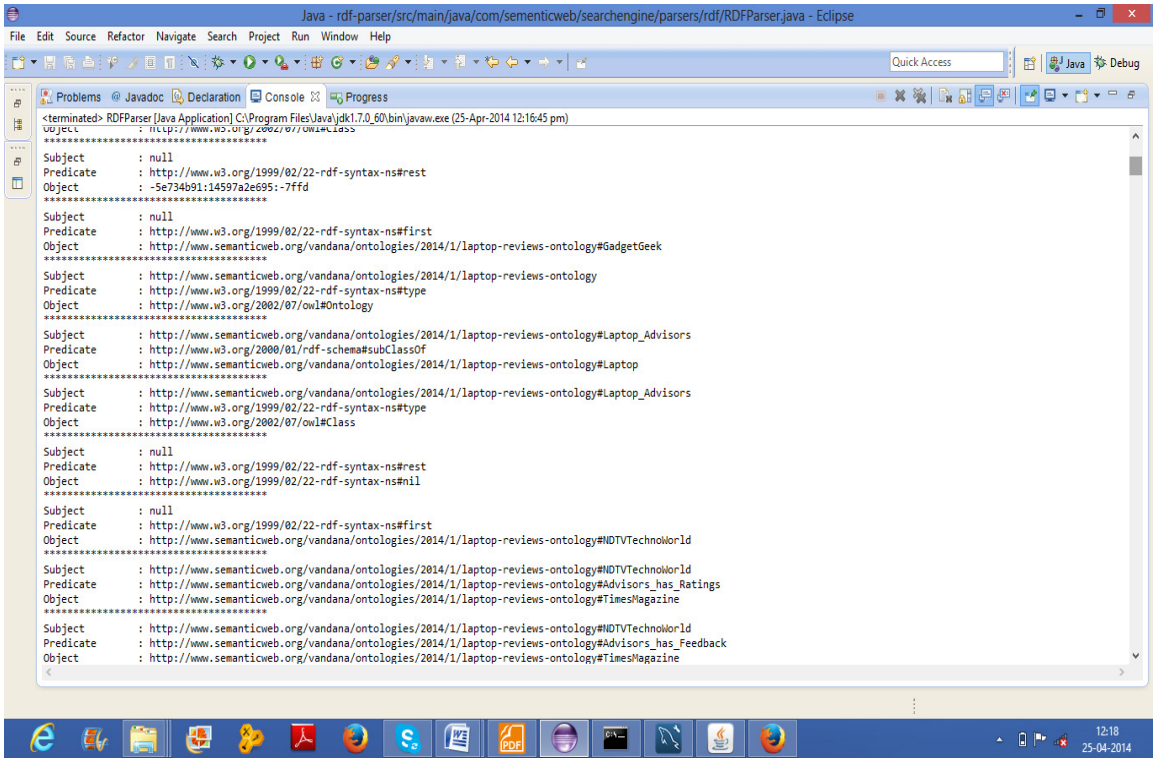

Figure 6.13 Parsing of results to <S, P, O>

Figure 6.13 depicts the parsed output of <S, P, O> done using Jena framework which describes the logical relations for RDF statements.

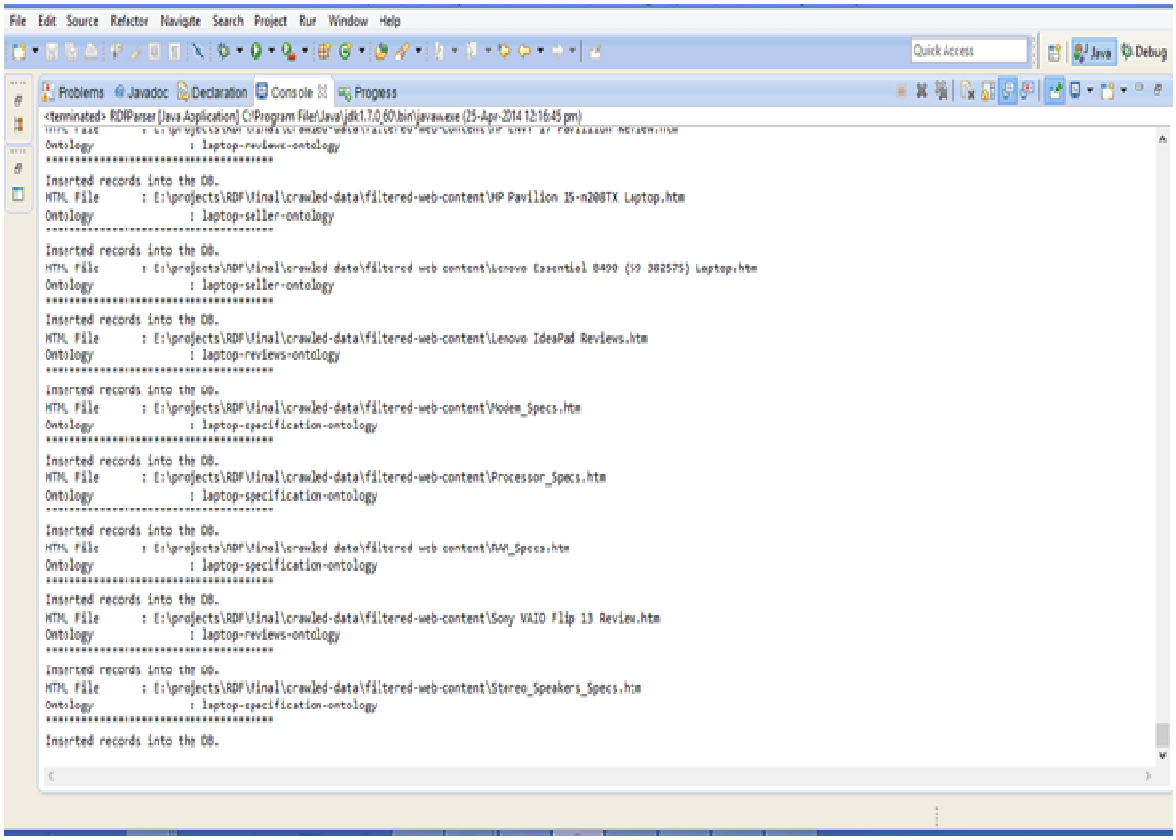

Figure 6.14 Ontology Extractor Output

Figure 6.14 depicts the <HTML file, Ontology > extraction, the output of Ontology Extractor Module which signifies the HTML file associated to ontology. Consider example in Table 6.2.

Table 6.2 HTML File and the associated Ontology

HTML File	**Associated Ontology**
E:\projects\RDF\Final\crawled-data\filtered-web-content\ Stereo_Speakers_Specs.htm	laptop-specification ontology

The HTML file and the associated Ontology for the parsed results are depicted in Table 6.2.

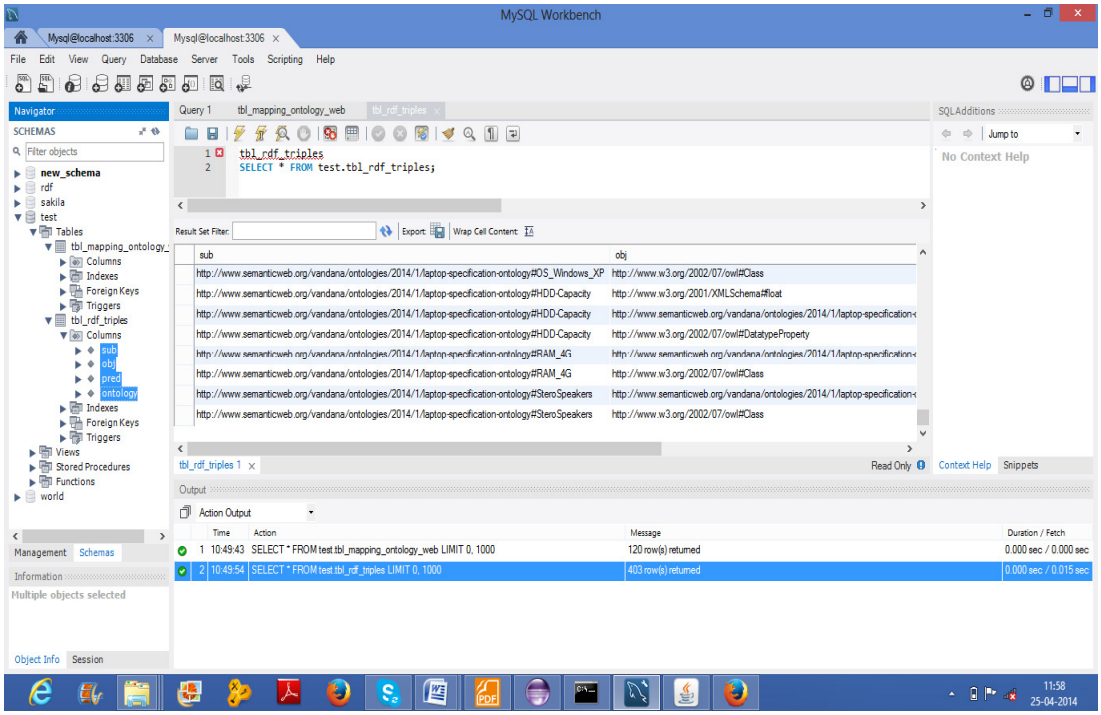

Figure 6.15 Ontology index of <Subject, Object, Predicate, Ontology>

Figure 6.15 depicts the ontology index of <Subject, Object, Predicate, Ontology> stored in MYSQL database. The <Subject, Object, Predicate> is parsed using Jena Parser and the index contains the associated ontology with the parsed triples.

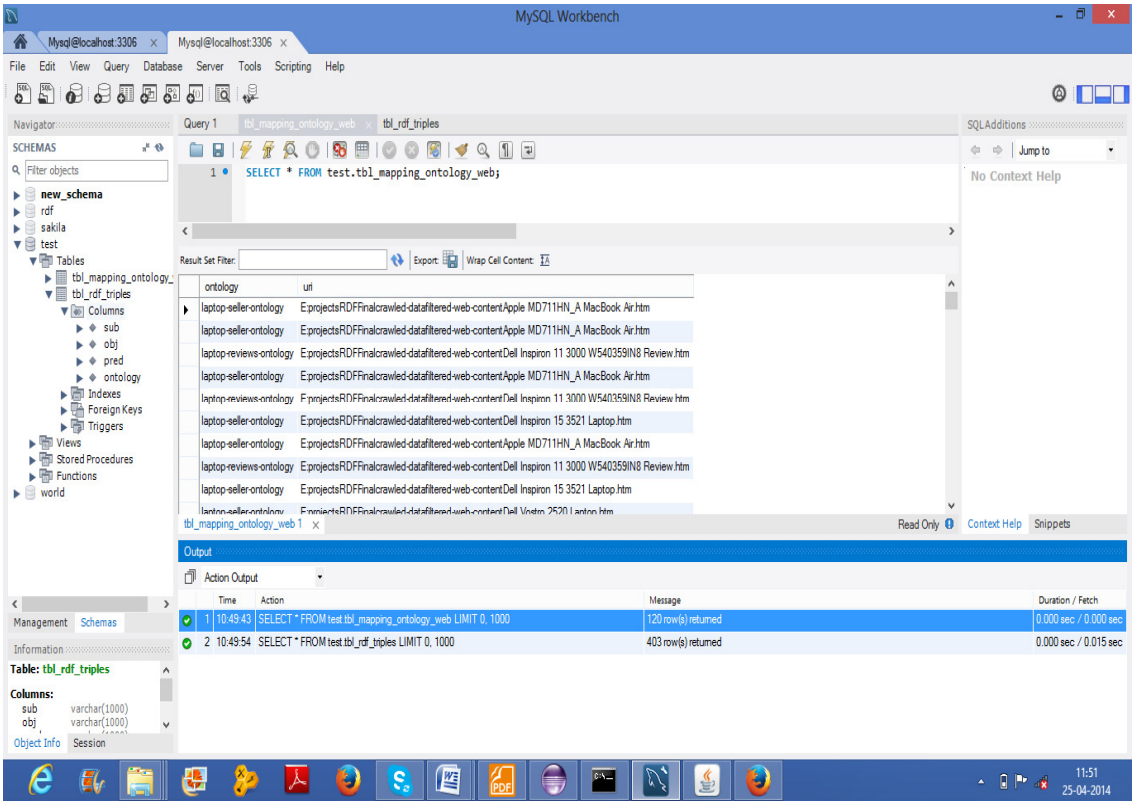

Figure 6.16 Index of < HTML file, Ontology, > in database

Figure 6.16 shows the index of <HTML file, ontology> into the MYSQL database which specifies HTML files which are associated with the ontology.

6.5.1 Evaluation Metrics

Two indexes, <S, P, O, Ontology> and <HTML file, Ontology> has been created. The proposed system states that rather than deploying full text indexation technique for web contents, it is better to index the ontology associated with pages. The proposed framework is advantageous in terms of index size in comparison to full text indexation. The below graph depicts the index size of web contents with only ontology indexed and index size of web contents with the index of the whole page. The first approach of indexing only ontology have index size small as compared to indexing the whole web contents. Figure 6.17 depicts the graph of web content vs. index size i.e. the HTML page indexing have index size more as compared to proposed search.

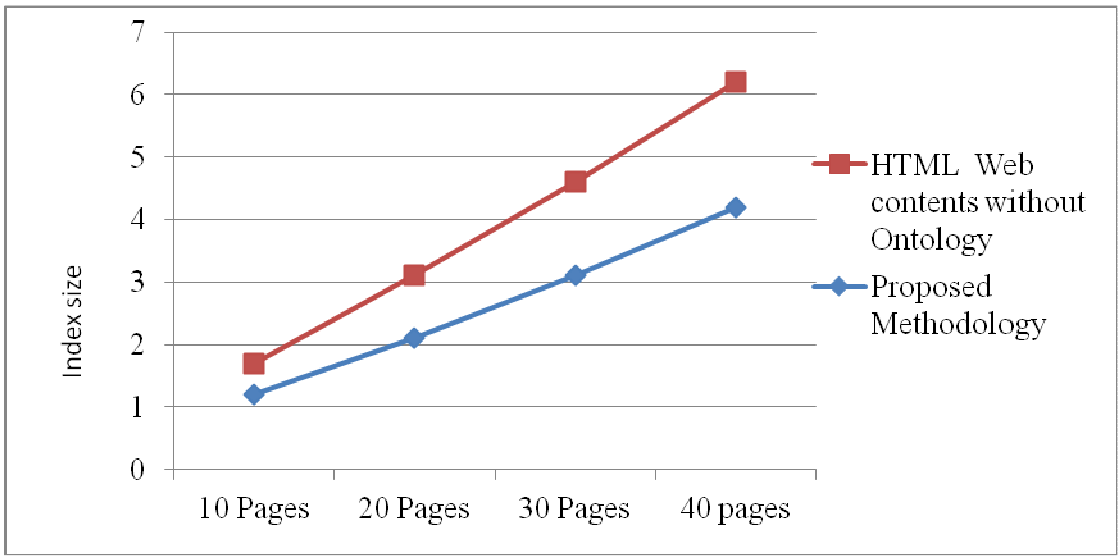

Figure 6.17 Graph of HTML web contents without Ontology Vs using SemIndex

The proposed architecture of SemIndex module was well implemented using MYSQL database for storing indexes and Jena for parsing and results shows that index size as compared without annotated ontology repositories have more space requirement as compared to the proposed approach and in addition the proposed approach well justifies the index creation for information retrieval.

CHAPTER VII

7. SEMRANK: A NOVEL HYBRID APPROACH FOR RANKING ONTOLOGY ANNOTATED WEB DOCUMENTS

7.1 INTRODUCTION

A large result set is generally produced in response to user query to the search engine and the user is forced to search through the complete set of results which in turn is a labour-intensive work. Hence, there is a need for ranking the result pages producing the more relevant results at top and less relevant results afterwards. Highly ranked pages have the probability of containing more relevant information as compared to low ranked page. Such a ranked system is called ranked retrieval system.

Google has Page Rank [42] algorithm that works on a set of simple web pages, Google ranking system is based on around 200 ranking features including in-links, out-links, personalization which gives a set of ranked results over a collection of web pages.

Google is the search engine that order its search results based on page's "popularity" as computed from the Web's graph structure [42]. But, Google's PageRank algorithm, [42,111,113] which is based on the "random surfer model", cannot be directly used in the Semantic Web as URIs mentioned in a RDF/OWL documents are not merely hyperlinks but semantic symbols referencing classes, semantic web instances, ontology documents, normal Web resources, etc. Semantic Web surfing is not merely random hyperlink-based surfing but rational surfing that requires understanding the semantic content of documents [167].

Therefore, the proposed approach for ranking of semantic documents is required to work on collection of web pages where knowledge representation techniques called ontologies has been used to annotate documents. The novel ranking approach is proposed which is hybrid approach, having two dimensions of ranking- ranking

metrics based on traditional model of web page ranking and ranking metrics for ontology annotated web pages.

7.2 COMPARATIVE ANALYSIS OF ONTOLOGY RANKING ALGORITHMS

For finding ontology various search engines like Swoogle, Falcon, and Watson are developed. These search engine returns a set of ontologies corresponding to a set of query in contrast to Google which is used to find web pages corresponding to a set of query. The searched ontologies with regard to a domain can be reused. For ranking these ontologies their has been various techniques- AKTive Rank, Swoogle Rank, Ontokhoj, SemSearch, Ontoselect, Falcon algorithm. Table 7.1 depicts a detailed comparison of ranking algorithms used for selecting ontology.

Table 7.1 Comparative Analysis of Ontology Ranking Algorithms

Ranking Algorithm	**Ranking Method**	**Ranking Technique Used**	**Limitations**
AKTiveRank [168,169]	Structure analysis of concepts.	AKTiveRank matches query terms with the labels of the classes in the ontologies. It uses four types of measures for each ontology to measure the ranking. The four measures it uses for ranking are Class Match Measure (*CMM*), Density Measure (*DEM*), Semantic Similarity Measure (*SSM*) and Betweenness Measure (*BEM*). It uses features of concepts such as their hierarchical centrality, structural density and semantic similarity.	**1**. Introducing these time- consuming metrics and being lack of efficient implementation [170] make AKTiveRank unsuitable for large scale semantic web. **2.** Also, the resulting score depends very much on user's queries. **3**. Individual instances and vocabulary terms cannot be ranked. **4**. The AKTiveRank cannot respond to the real-time queries as the final results cannot be reached until the preliminary results from the ontology search engine are analysed.
Swoogle [111,112,113]	Similar to the PageRank algorithm called as Rational random surfing model	1. Similar to the Page Rank algorithm with analysing the link structure. 2. Analyses links and referrals between the ontologies in the hope of identifying the most popular ones. 3. The assumption underlining their ranking methods is that there are hyperlinks existing in the ontologies on the SW to connect each other and the weight of an ontology can be determined by the number of	1. However, the majority of ontologies available on the Web are poorly connected, and more than half of them are not referred to by any other ontologies at all. Poor connectivity of ontologies would produce poor Page Rank results. 2. Furthermore, a popular ontology does not necessarily indicate a good representation of all the concepts it covers. Popularity does not necessarily correlate with 'good' or appropriate representations of knowledge. 3. Search does not allow properties of ontology such as structure to be searched.

		citations from other ontologies.	
Ontokhoj [108]	Similar to PageRank algorithm.	Ranking algorithm termed OntoRank which assigns a rank to an ontology in Semantic Web, based on priorities for different types of relationships.	Onto Rank algorithm does not take into account the correlation between the search result and user's query, which leads to ontology of low correlation posed in ranked result list.
OntoQA [172]	Based on certain metrics	OntoQA evaluates ontologies based on certain factors of schema and instances and overall score for ontology for ranking.	It has limitation of user involvement and has only keyword based query type.
OntoSearch [181]	Structure based	It is based on searching for structure using a simple query language which allows all the requirements identified to be covered.	It can only search for one type (RDFs) of ontology file, and it only compares the user keywords with the contents of the ontology files wherever they occur. And so it matches indiscriminately the keywords both from concepts and comment fields [181].
SemSearch [173]	Closeness to user specified queries	The search engine considers two factors while ranking - matching distance between each keyword and its semantic matches and the other is the number of keywords that match the search result.	SemSearch does not consider relations between concepts, and neither does it disambiguate in cases when there is more than one relation.
Recon Rank [174]	Uses PageRank /HITS algorithms	It uses PageRank type algorithm, which unifies the documents and resources in a dataset. The method generates scores for the documents and entities in a collection, but not for the properties.	Recon Rank does take data provenance into account, however because it simultaneously operates on the object graph, it is susceptible to spamming.
Ontoselect [175]	Structure based	This algorithm rank dynamic ontologies and is based on three measures and a combined score of that measures.	It lacks concept based searching of the ontologies and just matches the user entered query keywords with classes and properties in ontologies.
Falcon [115,116]	Uses popularity based ranking	It ranks concepts and ontology according to their relevance to the keyword query and their popularity on the Semantic web.	It has current limitation of user interaction and has only keyword based query type.

Related work discussed in Table 7.1 is based on ranking ontologies, and these approaches use various measures as discussed to compute the ranking while searching for ontology in a certain domain. Whereas the approach in the current research is to rank the web pages which are annotated with ontologies. Thus, it requires two dimensions for ranking-one is based on metrics for finding particular ontology that matches the search term of the query and the other dimension is to rank all the web pages associated with that ontology, after finding the particular ontology.

These ranking techniques covers one dimension of research -selecting ontology and for the other dimension traditional approach for ranking techniques was deployed.

7.3 OBJECTIVES OF PROPOSED SEMRANK

The objectives of the proposed research work are as follows:

1. Design of architecture for ranking semantic annotated web pages.
2. Proposing a ranking technique that supports two dimension of ranking
   2a. Ranking metrics for selecting particular ontology from the ontology corpus and
   2b. For the selected ontology, ranking the web pages annotated with the selected ontology.

The next section discusses proposed method for ranking of ontology annotated web documents.

7.4 PROPOSED ARCHITECTURE FOR RANKING ONTOLOGY ANNOTATED WEB DOCUMENTS

The architecture of SemRank has been proposed for ranking ontology annotated web documents as shown in Figure 7.1. It uses a mechanism for initially matching the concepts in ontology and then ranking the web pages annotated with that ontology.

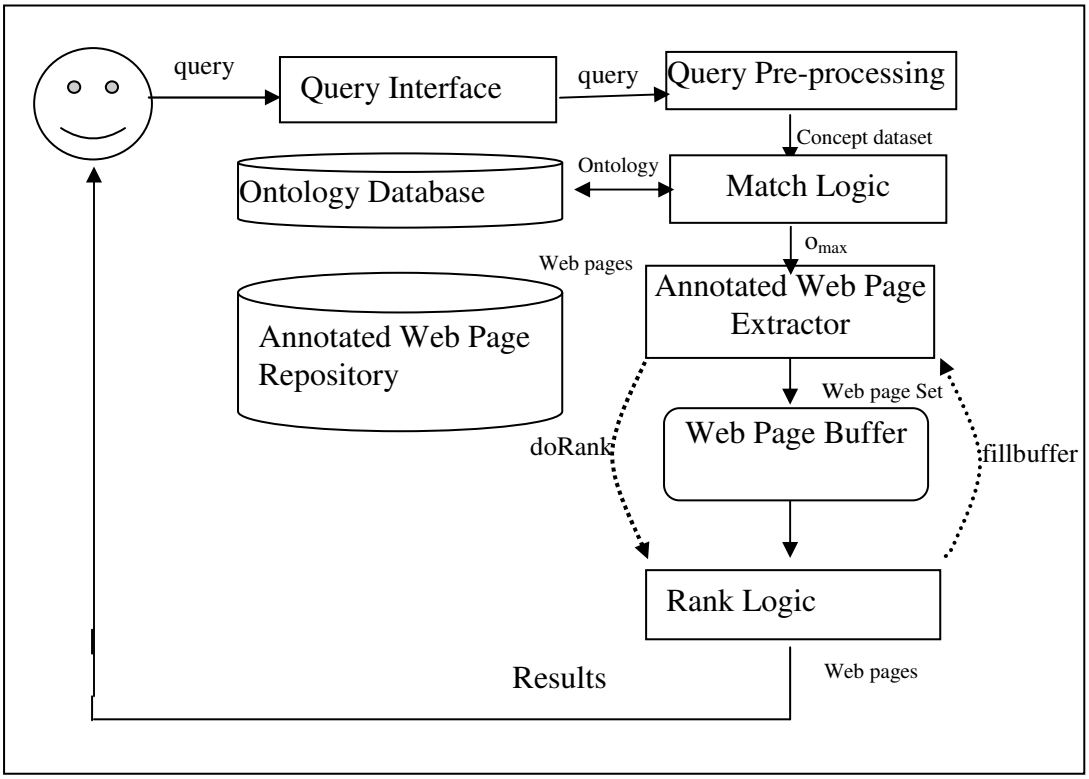

Figure 7.1 Proposed Framework for Ranking Ontology Annotated Web Pages

The proposed architecture consists of the following functional modules:

(i) Query Pre-processing

(ii) Match Logic

(iii) Annotated Web Link Extractor

(iv) Rank Logic

The user enters a query through Query Interface Module which is preprocessed by matching with WordNet [105] vocabulary. The dataset of concepts in query and WordNet matched vocabulary is given as input to Match Logic Module where using the ranking metrics *Match_Measure (O)*; the ontology with the maximum *Match_Measure*, O_{max} is found. The web pages linked with O_{max} ontology is extracted using Annotated Web Page Extractor Module; the web pages related to a ontology are ranked using Rank Logic Module that uses *Rank_Score* metrics. The ranked web page set is given as result to the user.

7.4.1 Query Pre-processing

The query entered by the user is checked with the WordNet [105] for synonym, hyponyms and meronyms of the original query term, all the data sets has been stored for matching with the concepts in the ontology. This adds to generalization of the query term. For example searching for query "laptop price" all other concepts like Laptop cost, Notebook cost, Notebook price are searched in Ontology for the concepts matching the WordNet.

7.4.2 Match Logic

For the query entered by user, the concepts and the WordNet vocabulary have been matched with the classes in the Ontology. The Match Logic Module uses *Match-Measure (O)* metrics for selecting Ontology O.

Match-Measure **Definition:** Let Q = {C_1, C_2…….C_n} where C_n represents the concepts in a query Q of user and M be the set of potential class labels obtained from the ontology and the match for the query against the Ontology concepts has been computed as defined in equation 7.1 as

$$Match\_Measure(O) = \max_{1 \le i \le n} [\alpha.(Concept\_Weight\_Measure) + \beta\, f(R)] \quad (7.1)$$

Where

i = number of ontologies

$f(R)$ = Function of Relatedness

α, β are parameters, where $\alpha + \beta = 1$; $\alpha \le 1$ and $\beta \le 1$

Concept_Weight_Measure is defined in equation 7.2 as

$$Concept\_Weight\_Measure = \sum_{i=1}^{n} f(M, Cn) \quad (7.2)$$

where

n = number of concept in query

$f(M, C_n)$ = function which gives value 1 if ontology contains a exact class label matching C_n and 0 if ontology does not contain a class label matching C_n.

$$\text{where } f(M, C_n) = \begin{cases} 1 \text{ if ontology contains exact class label matching } C_n \\ 0 \text{ if ontology does not contains a class label matching } C_n \end{cases} \quad (7.3)$$

Concept_Weight_Measure is based upon Class Match Measure (CMM) of AKTive Rank Algorithm [169].

Function of Relatedness f(R)-It is the other factor which has been used to select the ontology. Relatedness refers to, if a concept is related to other concept. For example for query "Laptop Specification Dell", the concept laptop specification and concept Dell if found to be in triples as <Subject, Property or Object> are assigned weight as 0 otherwise it is 1.
is as defined in equation 7.4 as -

$$f(R) = \begin{cases} 1 \text{ if ontology contains the concepts in any triple} \\ 0 \text{ if doesn't contains the concepts in triples} \end{cases} \quad (7.4)$$

In measure *Match_Measure(O)*, $\alpha = 0.6$ and $\beta = 0.4$; *Concept_Weight_Measure* is given more weightage than *Function of Relatedness*, as it is more important for concept to be found in ontology. The algorithm for selecting the ontology having the maximum *Match_Measure (O)* is as given in Figure 7.2.

```
Algorithm Match-Measure (O) (input: query (C1, C2…..Cn) ,  output: ontology with
                                        max Match_Measure, Omax )
{
for each concept C do
        for each ontology do
        {
           find Concept_Weight_Measure;
           add the concept_weight in array [weight, ontology];
           Calculate f(R);
        }
find Match-Measure for each ontology;
return ontology with max Match-Measure;
}
```

Figure 7.2 Algorithm for Match-Logic (O) of Ontology

7.4.3 Annotated Web Page Extractor

This module is responsible for extracting web pages related to the selected ontology. Ontology matching a query is selected by Match Logic with certain measures and all the web pages which are annotated with the selected ontology is extracted. All the extracted web pages are then stored in Web Page Buffer. The algorithm for Annotated Web Page Extractor is as described in Figure 7.3

```
Algorithm Extractor (W) (input: ontology, output: web pages)
{
Wait (fillbuffer)
for ontology with max Match_Measure(O), O_max do
  {
   find (O_max in web_page_repository);
   store all web pages corresponding to O_max;
   }
signal (doRank);
return (web pages);
}
```

Figure 7.3 Algorithm for Annotated Web Page Extractor

The input to Extractor algorithm is ontology and the output is a set of web pages corresponding to that ontology. It waits for signal *fillbuffer*, upon receiving the signal *fillbuffer* from Rank Logic Module it extracts the web pages corresponding to O_{max} and store those web pages in Web page Buffer and signal *doRank* signal to the Rank Logic Module.

7.4.4 Rank Logic

After finding the appropriate ontology, there is need to rank the web pages annotated with that ontology which is based on certain metrics. The web pages extracted for the selected ontology O_{max}, the web pages are ranked based on *Rank_Score* metrics.

The ***Rank_Score*** is specified in Equation 7.5 as

$$Rank\_Score = \alpha.(TextFreq(c,q)) + \beta.(Linkweight(PR(A))) \qquad (7.5)$$

Where $\alpha + \beta = 1$; $\alpha \leq 1$ and $\beta \leq 1$

The value of α=0.6 is considered giving more weightage to the similarity of the *TextFreq* and β=0.4 to *Linkweight.*

TextFreq(c, q) refers to the frequency of the concept similar to the query in web page. ***Linkweight(PR(A))*** refers to the weight measure of links from a given web page document A, A's Page Rank is computed as specified in equations 7.6, 7.7, 7.8 as

$$PR(A) = PR_{direct}(A) + PR_{link}(A) \tag{7.6}$$

$$PR_{direct}(A) = (1-d) \tag{7.7}$$

$$PR_{link}(A) = d\left(\frac{PR(T1)}{C(T1)} + ... + \frac{PR(Tn)}{C(Tn)}\right) \tag{7.8}$$

Where T_1,....., T_n are web documents that link to A; $C(T_i)$ is the total outlinks of T_i; and *d* is a damping factor, which is typically set to 0.85[42].

Algorithm for computation of *RankScore* for a query is as given in Figure 7.4

```
Algorithm Rank Logic Module (input: set of web-page; output: set of ranked
                                web pages)
{
Wait (doRank);
For (all web pages for Omax) do
{
    Calculate frequency for each term in document;
    Calculate linkweight for each web-page;
   Compute Rank_Score;
}
Arrange web-pages in decreasing order of rank-score;
return (result web pages);
Signal (fillbuffer);
}
```

Figure 7.4 Algorithm for Rank Logic Module

The module takes as input the set of web pages and produces a set of ranked web pages. Upon receiving the signal *doRank* from the Annotated Web Page Extractor Module this module computes *Rank_Score* metrics and the ranked web pages are

returned as result set signalling *fillbuffer* signal to Annotated Web Page Extractor Module for further storage of web pages in Web Page Buffer.

7.5 EXPERIMENTS AND EVALUATION

In this work, the web pages have been annotated with the ontology. For experimental evaluation, results of 20 pages have been shown. Three ontology have been developed, *Laptop_Specification*, *Laptop_Seller* and *Laptop_Review*, these three ontology are annotated with the web pages. For query-"Laptop Cost", the concept is searched in the WordNet for synonym, hypernym and these concepts are searched in the ontology and *Match_Measure* was calculated for the query.

Table 7.2 Query concept for *Laptop Cost*

Query Fired	**WordNet Vocabulary**
Laptop Cost	Laptop-Notebook, Computer
	Cost-Price, Toll

The different concepts for query "Laptop Cost" is searched in WordNet Vocabulary, which is shown in Table 7.2. In order to compute *Concept_Weight_Measure* defined in equation 7.2 is used and its calculated value is depicted in Table 7.3.

$$Concept\_Weight\_Measure = \sum_{i=1}^{n} f(M, Cn) \qquad (7.2)$$

Table 7.3 Concept_Weight_Measure for ontologies

Ontology	**Concept_Weight_Measure**
Laptop_Specification(O1)	2
Laptop_Seller(O2)	1
Laptop_Review(O3)	1

Table 7.3 shows the computed *Concept_Weight_Measure* for different ontology, where for each ontology, their *Concept_Weight_Measure* is calculated with respect to a concepts of query.

Measure for *Relatedness* is calculated according to equation 7.4

$$f(R) = \begin{cases} 1 \text{ if ontology contains the concepts in any triple} \\ 0 \text{ if doesn't contains the concepts in triples} \end{cases} \quad (7.4)$$

Table 7.4 Measure for Relatedness

Ontology	Relatedness
Laptop_Specification(O1)	1
Laptop_Seller(O2)	0
Laptop_Review(O3)	0

Table 7.4 shows the *Relatedness* value for different ontology. Measure for metrics *Match_Measure* is calculated according to equation 7.1 as depicted in table 7.5.

$$Match\_Measure = \max_{1 \le i \le n}[\alpha.(Concept\_Weight\_Measure) + \beta.f(R)] \quad (7.1)$$

Table 7.5 Measure for Match_Measure

Ontology	Match_Measure
Laptop_Specification(O1)	1.6
Laptop_Seller(O2)	0.6
Laptop_Review(O3)	0.6

Match_Measure for *Laptop_Specification*(O1) ontology is 1.6. As *Match-Measure* for ontology O1 is maximum, ontology O1 is selected. Now for this ontology all web pages annotated with this ontology are extracted. The web pages corresponding to ontology O_{max} are ranked based on *Rank_Score* metrics. Overall computed *Rank_ Score* as defined in equation 7.5 is calcualted in Table 7.6.

$$Rank - Score = \alpha(TextFreq(c,q)) + \beta(Linkweight) \quad (7.5)$$

Table 7.6 Overall computed Rank_Score

Document	Query Concept	Textfreq(c,q)	LinkWeight	Overall Computed Rank_Score
D1	C1	16		
	C2	11	0.43	16.372
D2	C1	9		
	C2	11	0.56	12.224
D3	C1	6		
	C2	3	0.42	5.568
D4	C1	9		
	C2	4	0.72	8.088
D5	C1	12		
	C2	10	0.18	13.272

Table 7.7 shows *Rank_Score* for web pages and their corresponding ranks.

Table 7.7 Rank Score for Web pages

Web Document	Rank-score	Rank
P1	16.372	1
P2	12.224	3
P3	5.568	5
P4	8.088	4
P5	13.272	2

The ranked results for the query "laptop display" is as given in Table 7.8, which was compared with human approach as, mentioned in Table 7.9.

Table 7.8 Ranked results for Query "laptop display"

1. *http://www.blog.laptopmag.com/laptop-buying-guide.htm*
2. *http://www.compreviews.about.com/od/video/a/Laptop-Display-And-Graphics-Guide.htm*
3. *http://www.gadgetsguru.in/laptop/index.htm*
4. *http://www.pcworld.com/article/187749/laptop_buying_guide_making_sense_of_the_specs.htm*
5. *http://www.wikihow.com/Choose-Which-Laptop-to-Buy.htm*

Table 7.9 Human Rank vs. Proposed Approach

Ranked Results	Human rank	Proposed Approach
http://www.blog.laptopmag.com/laptop-buying-guide.htm	4	1
http://www.compreviews.about.com/od/video/a/Laptop-Display-And-Graphics-Guide.htm	2	3
http://www.gadgetsguru.in/laptop/index.htm	4	5
http://www.pcworld.com/article/187749/laptop_buying_guide_making_sense_of_the_specs.htm	3	4
http://www.wikihow.com/Choose-Which-Laptop-to-Buy.htm	1	2

The values for human rank vs. proposed approach for the set of web pages are as depicted as graph in Figure 7.5.

Figure 7.5 Human approach vs. proposed approach

7.5.1 Validation of Proposed Work

Pearson product-moment correlation coefficient [169] has been used to calculate the relationship between human rank and the proposed approaches specified in equation

$$r_{xy} = \frac{\sum x_i y_i - n\bar{x}\bar{y}}{n s_x s_y} = \frac{n\sum x_i y_i - \sum x_i \sum y_i}{\sqrt{n\sum x_i^2 - (\sum x_i)^2}\sqrt{n\sum y_i^2 - (\sum y_i)^2}} \quad (7.9)$$

Where r_{xy}= relation coefficient between x and y

$x_1 \ldots x_n$= refers to dataset x containing n values

$y_1 \ldots y_n$=refers to dataset y containing n values

If the correlation coefficient is close to the proposed approach(r >0.5) means it has high collinear relationship between two different data sets. The Pearson Correlation coefficient for human rank vs. proposed approach comes out to be r = 0.6 which indicated a positive correlation between proposed approach and human rank.

Architecture for ranking annotated ontology documents has been proposed in this chapter which needs first to select the ontology and then rank the web pages annotated with that ontology. The proposed approach is a novel approach having two dimensions-one, selection of an appropriate ontology and second, ranking the web pages annotated with that ontology which provides a motivation towards knowledge representation with ontologies. The research approach is investigated with human knowledge engineers to compare with the proposed approach which has shown comparable results. In addition, these results are also validated using Pearson-Product Correlation Coefficient.

CHAPTER VIII

8. IMPLEMENTATION AND RESULTS OF PROPOSED SEMENGINE

8.1 INTRODUCTION

SemEngine has been proposed in this research work, that searches ontology annotated documents for which the ontologies have been developed in the domain of laptop and the web documents are annotated with those ontologies. The crawling system i.e. SemCrawl, for crawling the domain ontology and the annotated web pages, the indexing system i.e. SemIndex, for indexing ontology annotated web documents, the ranking system i.e. SemRank for ranking ontology annotated web pages, has been proposed and developed. The different modules for the developed SemEngine system are as follows:

1. Development of Ontology
2. SemCrawl system
3. SemIndex System
4. SemRank system

This chapter describes the experiments and results for the system.

8.2 PERFORMANCE METRICS

The performance metrics used for the research are as discussed below

PRECISION Precision (P) is the fraction of retrieved documents that are relevant as given in equation 8.1

$$\text{Precision} = \frac{\#\text{ (relevant items retrieved)}}{\#\text{ (retrieved items)}} \quad (8.1)$$

RECALL Recall(R) is the fraction of relevant documents that are retrieved as given in equation 8.2 as

$$\text{Recall} = \frac{\#\text{ (relevant items retrieved)}}{\#\text{ (relevant items)}} \quad (8.2)$$

F-MEASURE It is the weighted harmonic mean of Precision and recall as given in equation 8.3 as

$$\text{F-measure} = \frac{2PR}{P+R} \quad (8.3)$$

Where P stands for Precision and

R stands for Recall

The next section discusses about the experiments and results conducted in order to validate the proposed work in the thesis.

8.3 EXPERIMENTS AND RESULTS

The domain specific ontology creation can be done either through tools like Protégé, OntoEdit or searching of required ontology can be done through various search engines tools like Swoogle, Watson, Falcon. For this research work, ontology was searched in the domain of laptop, but the relevant ontology with the required concepts was not available, therefore the ontology creation through ontology development tools was opted. Various ontology development tools are available for ontology development. For this research, **Protégé** framework has been used for ontology development.

The requirements for this approach to be implemented is that the pages are required to be annotated with relevant ontology which requires lot of efforts which can be done through various approaches like manual annotation, semi-automatic approach and automatic approach. In this research, ontology has been manually annotated as the other approaches require an intensive knowledge of machine-learning approaches and natural language processing techniques.

8.3.1 Development of Ontology

The following three ontologies have been developed in the domain of laptop using Protégé framework.

- *Laptop_Specification Ontology*

- *Laptop_Seller Ontology*
- *Laptop_Review Ontology*
- *Laptop_Specification* ontology in Protégé Framework has been shown in Figure 8.1.

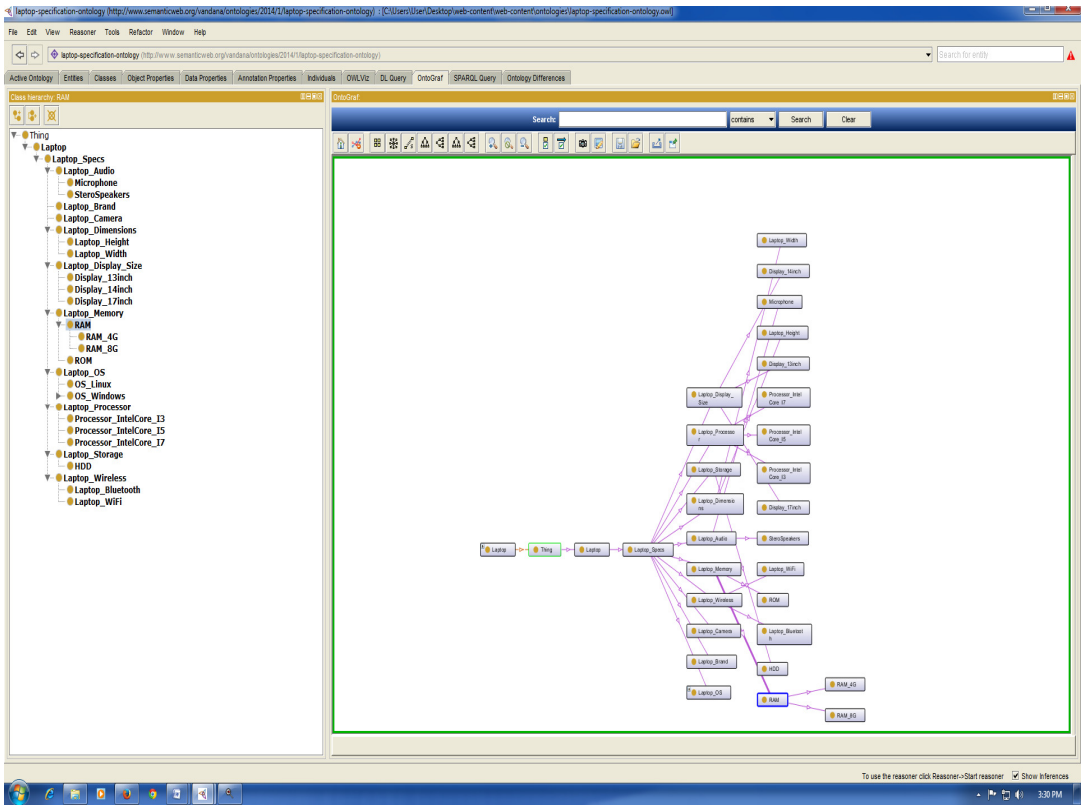

Figure 8.1 Development of Laptop_Specification Ontology in Protégé Framework

Figure 8.1 shows the development of *Laptop_Specification* Ontology in Protégé, the right side of the pane shows the different classes and subclasses of this ontology and the left side of the pane shows the ontovisualizer result of the developed ontology. The detailed implementation of the development of three ontologies has been discussed in **Chapter 4.**

Various experiments were conducted on 50 pages. For clarity, 20 pages with the associated ontology and plain HTML pages without any annotated ontology are as shown in Table 8.1.

Table 8.1 shows, five examples of web pages associated with *Laptop_Seller* ontology, five web pages annotated with *Laptop_Specification* ontology and five web pages annotated with *Laptop_Review* Ontology and five pages were not annotated with any ontology.

Table 8.1 Web Pages with the associated ontology

Specification	**Seller**	**Plain HTML Pages**	**Review**
http://www.pcworld.com/article/187749/laptop_buying_guide_making_sense_of_the_specs.htm	http://www.snapdeal.com/offers/reglobe	http://wikipedia.com	http://www.notebookreview.com/best_laptops/
http://www.compreviews.about.com/od/video/a/Laptop-Display-And-Graphics-Guide.htm	http://www.gadgetguru.in.laptop/index.htm	http://philapedia_Mint.com	http://www.pcadvisor.co.uk/reviews/laptops/5/
http://www.gadgetsguru.in/laptop/index.htm	http://www.cromaretail.com/Laptops-c-20.aspx	http://indiatravelguide.com	http://www.pcadvisor.co.uk/advisor/laptop/
http://www.blog.laptopmag.com/laptop-buying-guide.htm	http://www.junglee.com/Laptops/b/803540031	http://nanital Travel guide.htm	http://gadgets.ndtv.com/laptops/reviews
http://www.wikihow.com/Choose-Which-Laptop-to-Buy.htm	http://www.flipkart.com/computers/laptop.htm	http://aam_adamiparty.htm	http://sony.vaioFlip13review.htm

For Implementation, **Java** [29] with **Jena APIs** [154] is used in **Eclipse Development Framework** [29]. Jena Framework is designed and implemented by HP Labs; the Jena Framework is a set of Java APIs used for Semantic Web application development. Jena provides a reasoning subsystem for Ontologies developed using RDF, OWL.

8.3.2 SemCrawl Module

SemCrawl Module crawls the web documents from web, filters the web pages which are not annotated with ont2logy and parse those filtered web documents into <S, P, O> format for inferential support.

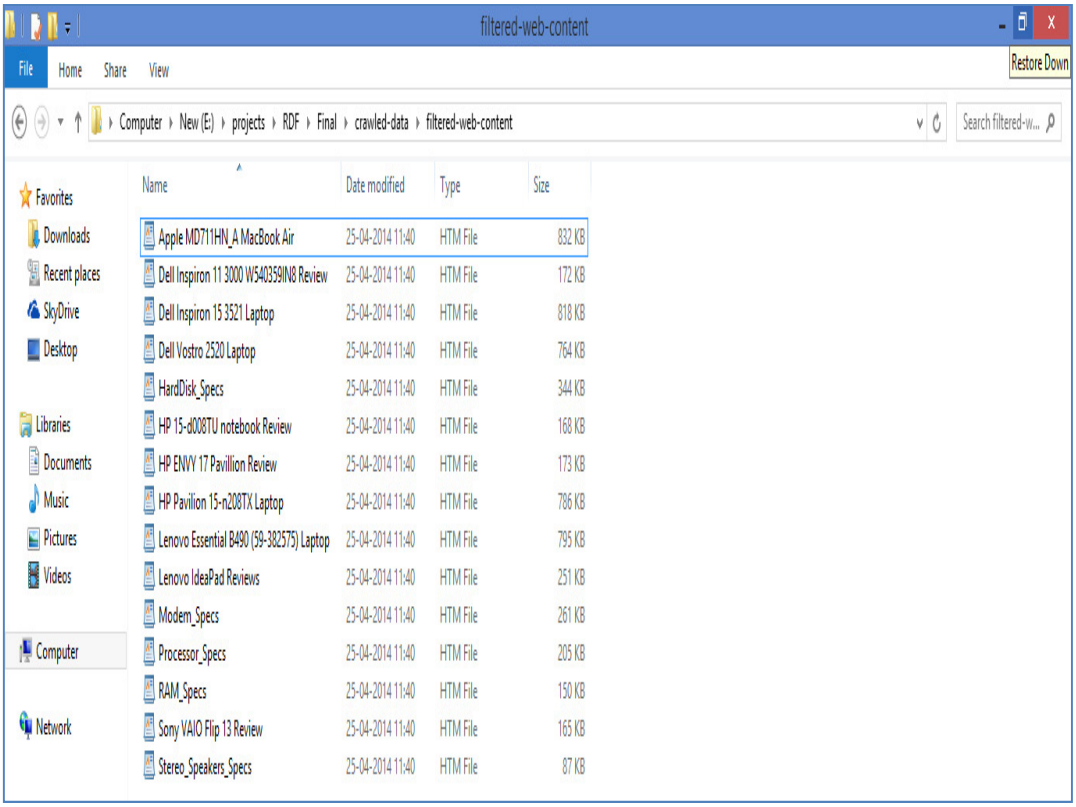

Figure 8.2 Crawler Module Result

The Ontologies created were annotated with various pages and then using SemCrawl framework was crawled. The implementation results of crawling for SemCrawl Module are as shown in Figure 8.2. The results contain the crawled web pages which are annotated with ontology and the plain HTML pages are filtered out. The details of implementation of SemCrawl have been discussed in detail in **Chapter 5.**

8.3.3 SemIndex Module

SemIndex Module then indexes the crawled ontologies and semantically annotated web pages. One index consists of<Subject, Predicate, Object, Ontology>, and the

other index is of< HTML file, Ontology>. Figure 8.3 shows the output of parsed results of ontology.

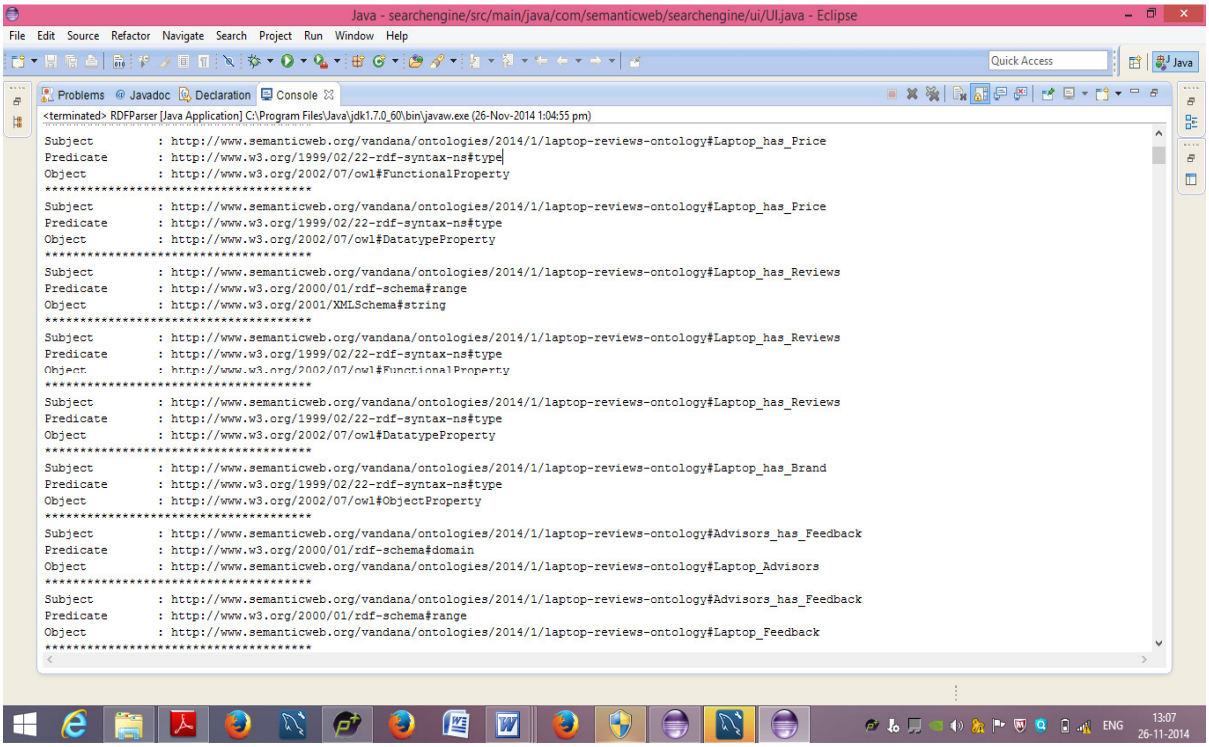

Figure 8.3 Ontology Parsing Result

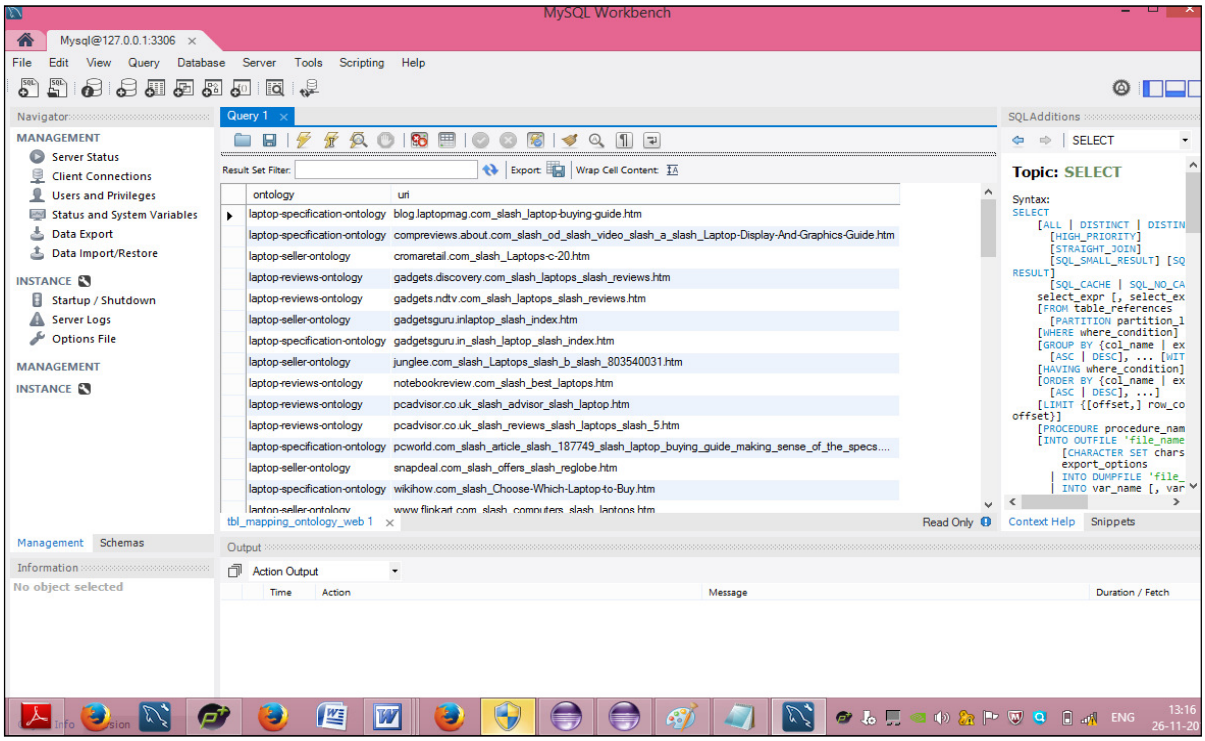

Figure 8.4 Ontology Index

The output of parsing result of ontology into <S, P, O> is shown in Figure 8.4 which is a triple relation for a RDF statement indicating the relation between the three entities. The index created by SemIndex module has been shown in Figure 8.4. These indexes were stored in **MYSQL database** for storage. The DL queries were executed in Protégé for checking the consistency of the developed ontologies as depicted in Figure 8.5.

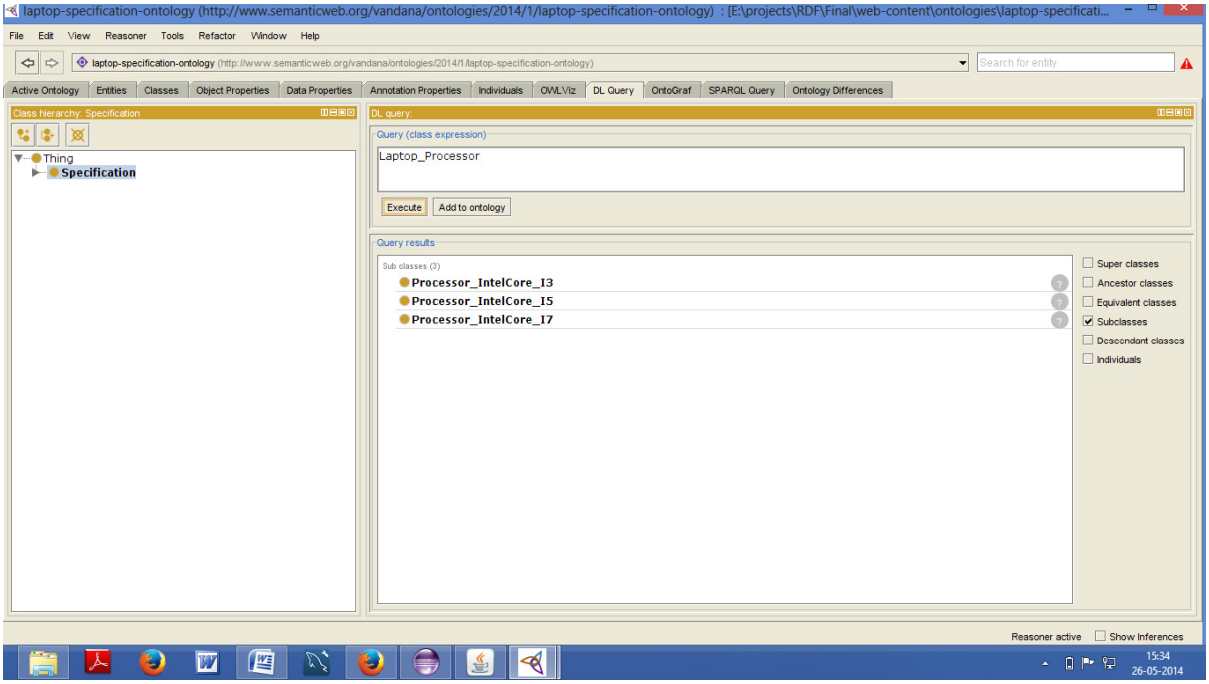

Figure 8.5 Execution of Query in Protégé

Figure 8.5 shows the execution of the query "Laptop_Processor" using DL Logic Query for *Laptop_Specification* ontology. The output of query is the subclasses of *Laptop_Processor* class. The details of SemIndex module has been discussed in detail in **Chapter 6.**

8.3.4 SemEngine

SemEngine searches results based on the ontology. A set of example queries were formed to analyze the results depicted in Table 8.2

Table 8.2 Set of evaluation query

Evaluation Query	
Query 1	Modem Specification
Query2	Laptop seller sites
Query 3	Laptop OS
Query 4	Laptop Specification

Table 8.2 shows the set of query executed on the proposed Search System. The proposed SemEngine then searches the appropriate web pages annotated with ontology concepts related to query. Figure 8.6 shows the interface for the developed SemEngine system which is divided into three subsections; Specification Results, Seller Results, Reviews Results (subsection name synonym to ontology name), which indicates that for the query entered, if it matches the concepts related to the developed ontology, the web pages associated with that ontology, are shown in that subsection.

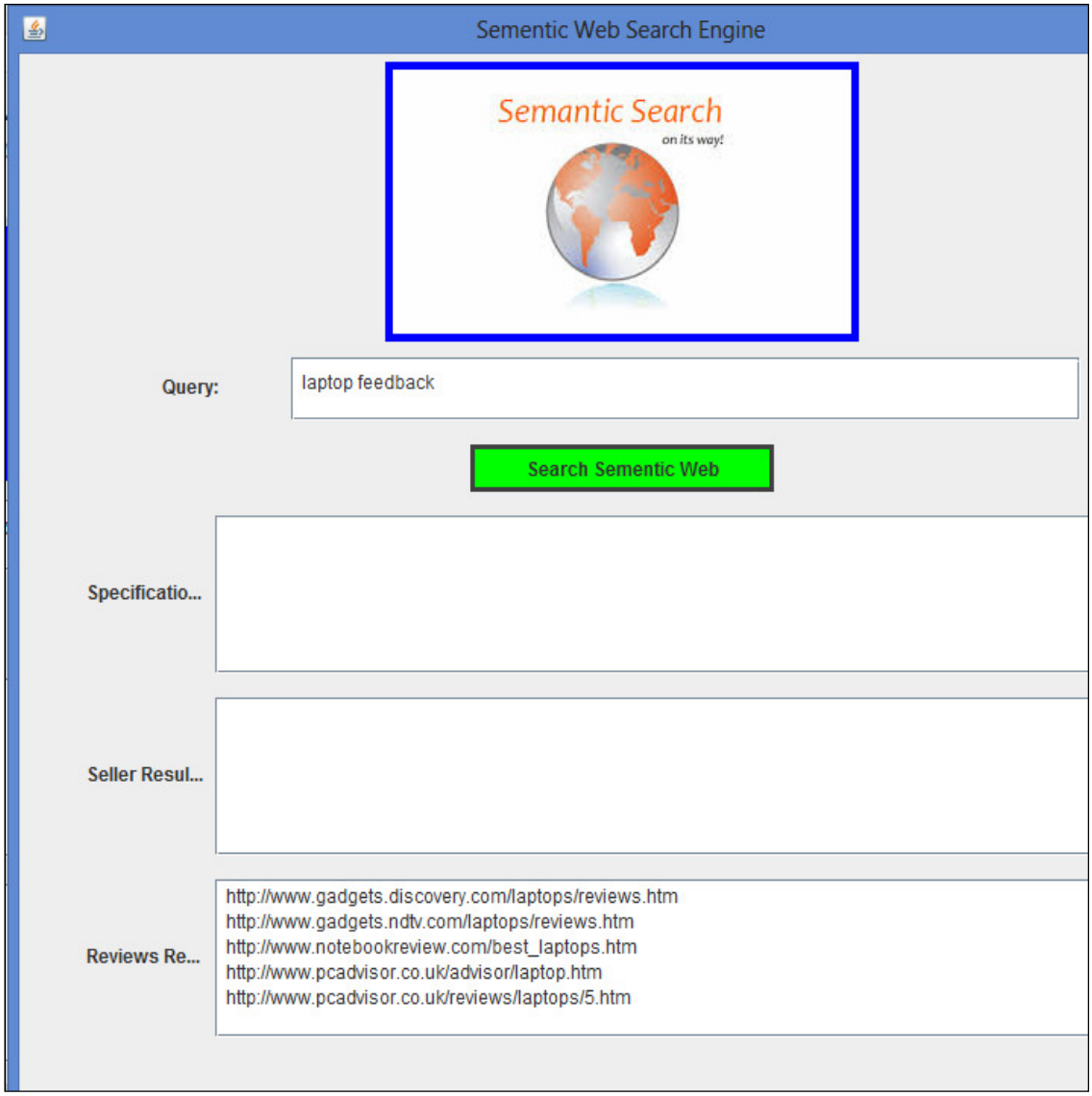

Figure 8.6 SemEngine Result Output

The output for the query "laptop feedback" that retrieves all the resultant web pages related to the *Laptop_Review* ontology has been shown in Figure 8.6.

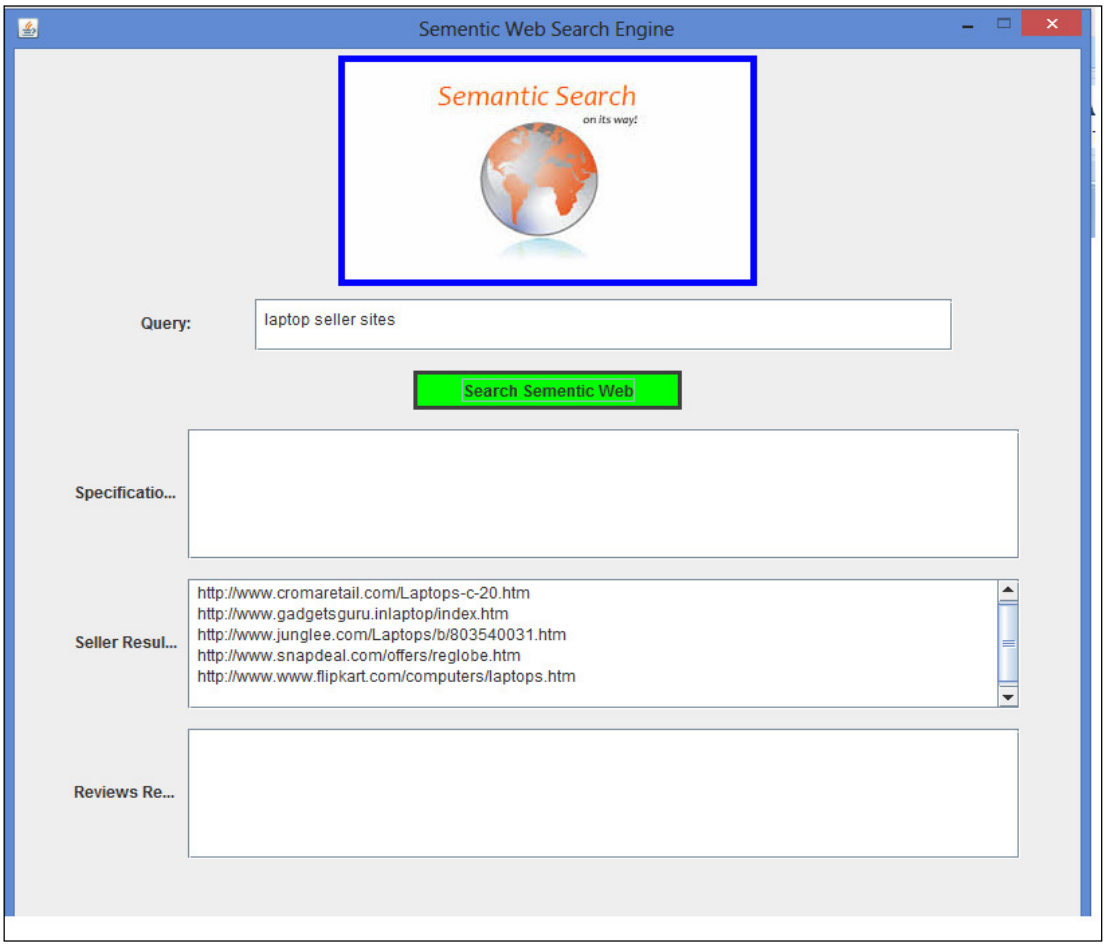

Figure 8.7 Semengine result for query"laptop seller sites"

The output for the query "laptop Seller Sites" that retreives all the web pages which relates to *Laptop_ selle r*ontology has been shown in Figure 8.7.

This approach is advantageous in comparison to keyword based approaches in terms of

- Semantically enriched information
- Index size less in comparison of HTML indexing where whole HTML page is to be indexed

The proposed approach is different from other approaches in the following aspects:

- There is no need for parsing the complete HTML page
- There is no need for creating Index for the whole HTML page as usually is done with traditional search engine techniques [38].

- Unlike general Search Engines search results are based on the concepts contained in the ontology rather than the keywords of the HTML page.

8.4 PERFORMACE EVALUATION OF THE SYSTEM SEMENGINE VS GOOGLE

For the same set of queries, the performance of SemEngine has been compared with the performance of Google. Both the systems return a list of web pages in reference to a particular query. The difference between both systems is the representation techniques of the web pages. Google finds the irrelevant web pages as the result whereas SemEngine searches for a set of web pages which are annotated with semantic languages which well represents a web page. The system was compared for a set of 50 pages in the terms of the results returned on the basis of the evaluation metrics which are widely used in the field of information retrieval known as Precision, Recall and F-measure.

Table 8.3 Experimental Evaluation of the System

Google		**SemEngine**					
Evaluation	**Google Precision**	**Ideal Precision**	**Actual Precision**	**Ideal Recall**	**Actual Recall**	**Ideal F-measure**	**Actual F-measure**
Laptop Feedback	67%	100%	86%	100%	88%	100%	87%
Laptop Review	60%	100%	82%	100%	92%	100%	87%
Laptop Seller sites	80%	100%	92%	100%	90%	100%	91%
Laptop Memory	53%	100%	94%	100%	88%	100%	91%

Table 8.3 shows the comparison of Google with the SemEngine. In ideal situation, if the ontologies are annotated with the web page correctly, the performance is 100%. In proposed research work, the ontologies are annotated with the web pages, but the web pages may or may not be always annotated correctly. Due to this reason, the actual performance is less than the performance in ideal situation.

The precision curve for the SemEngine vs Google is as shown Figure 8.8.

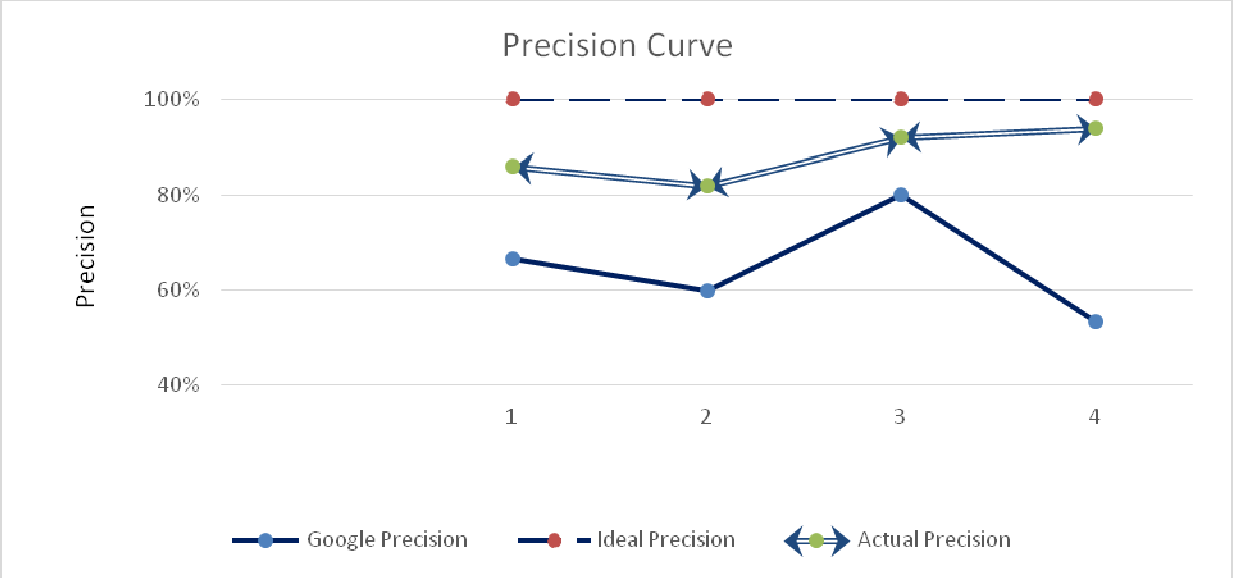

Figure 8.8 SemEngine Precision Curve vs Google

Figure 8.8 shows a graph between Google vs SemEngine results for the Precision metrics. The graph shows that in ideal situations SemEngine curve is 100% whereas actual precision depends on the annotations of the web pages with the ontology and is improved then the precision of Google. The ideal SemEngine system wherein the ontologies have all correct annotations; the curve for it is straight line.

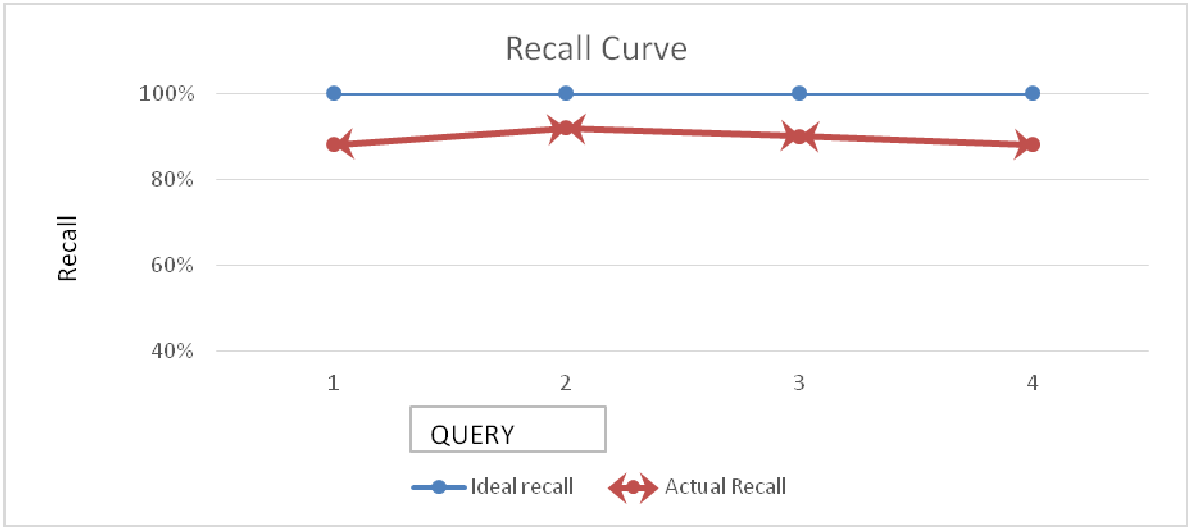

Figure 8.9 SemEngine Curve of Ideal vs Actual Recall

Figure 8.9 shows Ideal vs Actual recall value for SemEngine. In ideal situation SemEngine have recall value 100% for the query specified whereas in actual the curve varies and depends on the annotated web page.

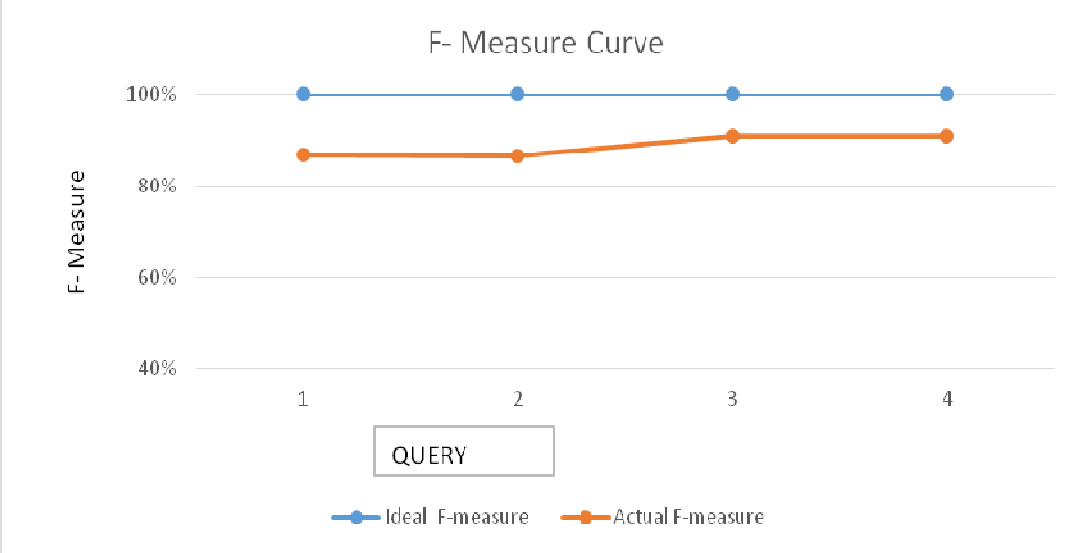

Figure 8.10 SemEngine F-Measure Ideal vs Actual Curve

Figure 8.10 represents the F-measure metrics curve for SemEngine in ideal vs actual value which represents the symmetric curve.

The results for the developed system has better precision as compared to Google but the performance of the developed system depends on certain factors-

- Coverage of the ontology concepts
- Scope of the ontology
- Proper annotation of the web pages with the relevant ontology

To validate the system support scalability, the numbers of queries were increased and the graph was plotted for precision depicted in Figure 8.11

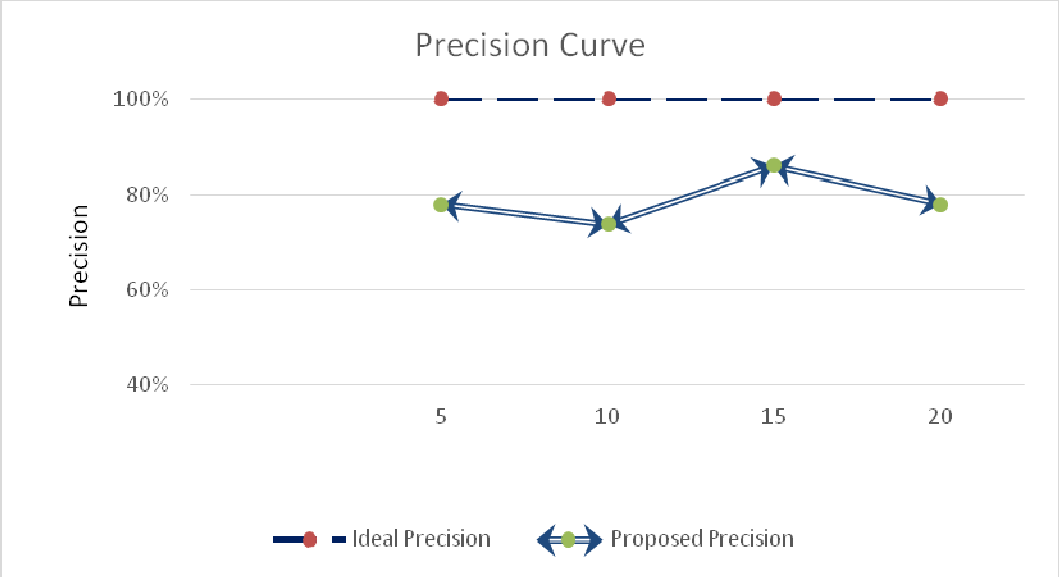

Figure 8.11 Precision Graph for the increased number of queries

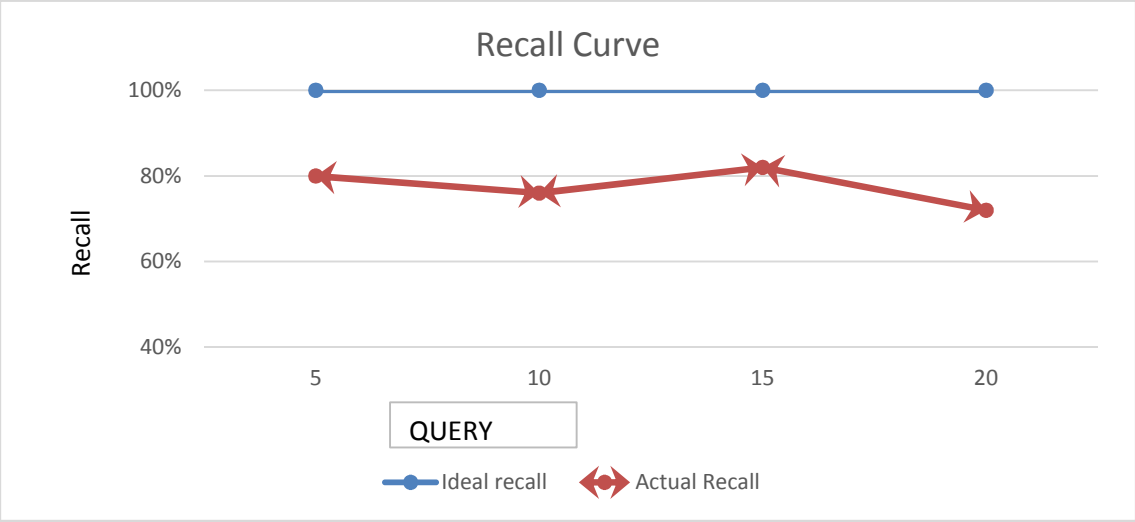

Figure 8.12 Recall Graph for the increased number of queries

With the increased number of queries Figure 8.11 and Figure 8.12 shows the precision and recall curve, for ideal vs. actual system, where ideal refers to system where all ontologies have been correctly annotated. As the number of queries were increased the values of precision and recall is symmetric and not that varying which indicates that system supports scalability.

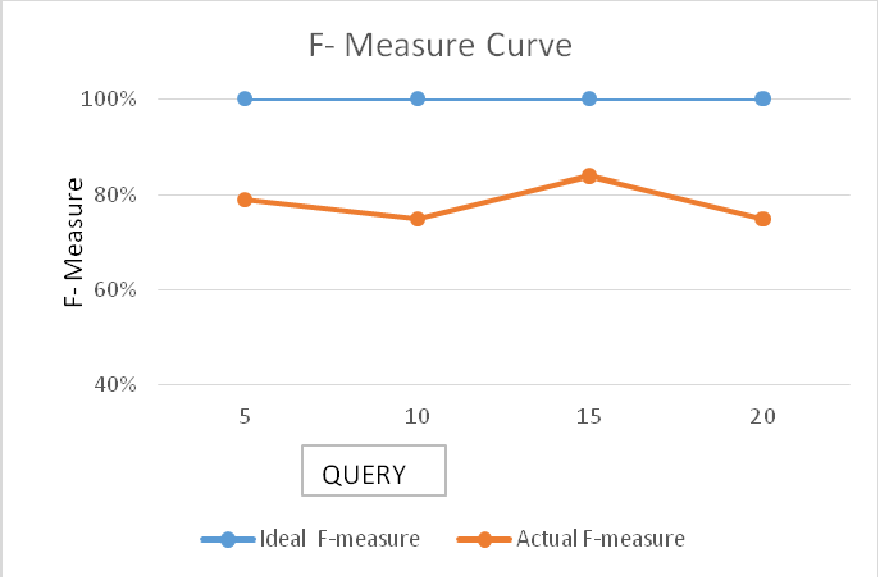

Figure 8.13 F-measure graph for the increased number of queries

Figure 8.13 indicates the F-measure curve for SemEngine when the numbers of queries were increased as indicated, SemEngine functionality does not get affected by increasing the number of queries.

Google is a general purpose search engine which fetches the simple HTML pages from the web and produces results based on the keyword matching whereas the developed SemEngine system fetches, crawls and indexes ontology annotated documents. The results shows the developed system shows the relevant results but the performance of the system depends on the above mentioned factors.

Comparative analysis of proposed SemEngine has been compared with Search engine as indicated in Table 8.3

Table 8.4 Comparative analysis of Search Engines

Features	**Swoogle**	**Falcon**	**Google**	**SemEngine**
Description	Based on the user query, the concept is searched in the repository and the ranked ontologies are returned as result.	Based on the keyword in the query the concept is searched in the repository and matching ontologies are	Based on the user query the keyword is matched in the web page and the ranked web pages are returned as	Based on the user query the concept is searched in ontology annotated web page and the ranked web pages

		returned.	result.	are returned as result.
Results	Ontology	Ontology	Web pages	Web Pages
Crawl Mechanism	Crawl and discover document written in RDF, OWL. Uses three crawlers • Focussed crawler (for searching documents within a given website). • Google crawler (for searching .rdf, .owl files). • Swoogle crawler to explore semantic link between SWDs.	Crawl and discover document written in RDF, OWL. Uses RDF Crawler.	Crawl and discover document written in HTML, XML. Uses Crawler "Googlebot".	Crawl and discover document written HTML associated with ontology. Uses SemCrawl
Ranking Mechanism	OntoRank Algorithm	RankingScore[116]	PageRank Algorithm	SemRank Algorithm

In this chapter the proposed SemEngine implemented in Java framework has been discussed. Jena framework is used for parsing semantic web documents to inference the underlying knowledge of the represented ontology.

CHAPTER IX

9. CONCLUSION AND FUTURE WORK

9.1 CONCLUSION

SemEngine search system has been developed which searches the web contents based on the concepts represented in them by ontologies. Ontologies have been developed in the domain of laptop. The developed ontologies have been annotated with the relevant web pages. An architecture SemCrawl has been developed to crawl the web pages and filter the web pages which have no semantic annotation to it. An architecture SemIndex has been proposed to index the crawled repository which create two different indexes - <Subject, Predicate, Object, Ontology> and <HTML File, Ontology>. A framework SemRank has been proposed to rank the ontology and the associated web pages using *Rank_Measure* and *Match_Measure*. When the user enters the query the ranked ontology annotated web documents are returned as result.

Following are the milestones that are achieved in this dissertation:

1. **Development of Ontology**

Ontologies were developed using Protégé development framework following the complete steps of ontology development lifecycle. Three ontologies were developed in the domain of laptop; i.e. *Laptop_Seller, Laptop_Specification, Laptop_Feedback* ontology.

2. **Framework for Crawling System**

A framework for crawling system; SemCrawl, has been proposed in order to crawl the web pages and to filter out the pages which are not annotated with ontology.

3. Framework for Indexing System

SemIndex framework has been proposed for indexing the web pages which are annotated with ontology. The index created requires index size less in comparison to the index for the traditional indexing mechanism. The two indexes <S, P, O, Ontology> and <HTML file, Ontology> were created which helps in finding the relevant web pages related to the query.

4. Framework for Ranking Systems

A framework for ranking mechanism i.e. SemRank has been proposed to rank the web pages based on two dimensions of ranking. First dimension deals with first selecting the ontology which matches the query concepts; second dimension deals with ranking all the web pages related to the selected ontology.

5. Framework for Search System

A framework SemEngine has been proposed to list the ranked search results based on query.

The objectives proposed were achieved. Domain specific approach has been followed for the development of ontology. Concept based crawling, indexing and ranking has been used for developing the system. In order to ensure the practical implications the developed system, SemEngine supports the following features:

- **Scalability-** The concepts can be added to developed ontologies which increase the vocabulary of the ontology supporting scalability feature for the developed system.
- **Extensibility-**The developed system is extensible as any third party software can be easily incorporated.
- **Relevancy-** The results are relevant in order to fulfil the user requirement.
- **Robustness-** Ontologies are so designed that ontology revision will not change the foundedness of the resources that commit to an earlier version of the ontology.

- **Improved Evaluation Metrics-** It has been observed that if the pages are annotated correctly the performance of the proposed system is very high.

The system has been implemented in Java using Eclipse Framework for project development. Jena Inference engine was used to infer the relationships for the RDF model of the represented ontology. For indexing the indexed data was stored in MYSQL server.

The next section discusses the future scope of the proposed work.

9.2 FUTURE SCOPE

In this thesis, a search engine for ontologies has been designed and implemented that includes crawling, indexing and ranking for annotated web pages. Some of the possible extensions that can done in the future in this area are as follows:

1. **Automatic Annotation of web pages**

The web pages were annotated with manual annotation method. The research work can be carried in this direction for automatic annotation of web pages, although a lot of automatic annotation tools like Magpie and SHOE extension languages are available which have not proven to be that useful in the area of annotation.

2. **Recommender System**

Research can be pursued in the direction of development of Recommender system for E-commerce application and can be compared with the traditional system.

3. **Natural Language Query**

Research can be done for developing a system which can process the natural language query by matching with the concepts in the ontology.

4. Ontology Library

Ontology library can be developed for the different applications and provided as a web service which can be reused by applications as it is or can be extended. Currently different systems are there for ontology reuse but that are not that user friendly.

5. Ontology classification

Different Ontology can be developed and research can be done to classify the ontology into different domains and can be made available as ontology library.

REFERENCES

[1] T. Berners-Lee and M. Fischetti, *Weaving the Web: The original design and ultimate destiny of the World Wide Web by its inventor*, HarperInformation, 2000.

[2] V. Dhingra and K. K. Bhatia, "Towards Intelligent Information Retrieval on web," in *International Journal of Computer Science and Engineering*, Vol. 3, no. 4, April 2011.

[3] T. A. Brooks, "Web search: how the Web has changed information retrieval," *Information Research,* Vol. 8, no. 3, 2003.

[4] C. Manning, P. Raghvan, and H. Schütze. *Introduction to Information Retrieval*, Cambridge University Press, 2008.

[5] K. S. Candan, H. Liu, and R. Suvarna, "Resource description framework: metadata and its applications," *ACM SIGKDD Explorations Newsletter,* Vol. 3, no. 1, 2001, pp. 6-19.

[6] M. Kobayashi and K. Takeda, "Information retrieval on the web," *ACM Computing Surveys (CSUR), Vol.* 32, no. 2, 2000, pp. 144-173.

[7] A. Arasu , J. Cho, H. Garcia-Molina, A. Paepcke, and S. Raghavan, "Searching the web," *ACM Transactions on Internet Technology (TOIT),* Vol.1, no. 1, 2001.

[8] A. Heydon and M. Najork, "Mercator: A scalable, extensible web crawler," *World Wide Web*, Vol. 2, no. 4 , 1999, pp. 219-229.

[9] D. Sullivan, "How search engines work," *SEARCH ENGINE WATCH, at http://www. searchenginewatch. com/webmasters/work. html),on file with the New York University Journal of Legislation and Public Policy,* 2002.

[10] G. Antoniou and F. van Harmelen. *A semantic web primer*. MIT press, 2004.

[11] M. C. Daconta, L. J. Obrst, K. T. Smith. *A Guide to Future of XML, Web Services and Knowledge Management*. Wiley Publications, 2003.

[12] G. Sudeepthi, G. Anuradha and M. S. P. Babu, "A survey on Semantic Web Search Engines," *International Journal of Computer Science Issues*, Vol. 9, issue 2, no. 1, March 2012

[13] M. Wilson and B. Matthews, "The semantic Web: prospects and challenges," in *7th International Baltic Conference on Databases and Information Systems, 2006*, pp. 26-29.

[14] F. van Harmelen, "The semantic web: What, why, how, and when," in *IEEE Distributed Systems Online,* vol. 5, no. 3, 2004.

[15] G. Madhu, A. Govardhan, and T. V. Rajinikanth, "Intelligent semantic web search engines: a brief survey," *International Journal of Web & Semantic Technology (IJWesT),* Vol.2, no.1, January 2011.

[16] K. Breitman, M. A. Casanova and W. Truszkowski, *Semantic Web: Concepts, Technologies and Applications: Concepts, Technologies and Applications*. Springer Science & Business Media, 2007.

[17] N. Shadbolt, W. Hall, and T. Berners-Lee, "The semantic web revisited," *Intelligent Systems, IEEE,* Vol. 21, no. 3, 2006, pp. 96-101.

[18] K. Janowicz and P. Hitzler, "Semantic Search on the Web," IOS Press, Journal Interoperability, Usability, Applicability, 2010, pp. 1-7.

[19] V. R. Benjamins, J. Contreras, O. Corcho and A. Gomez-Perez, "Six challenges for the Semantic Web," In *workshop KRR*, 2002.

[20] J. Euzenat, "Research challenges and perspectives of the Semantic Web," *Intelligent Systems, IEEE* , Vol.17, no. 5 ,2002, pp. 86-88.

[21] L. W. Lacy. *OWL: Representing information using the web ontology language*. Trafford Publishing, 2005.

[22] T. Finin, A. Joshi, and V. Doshi, "Swoogle: A Semantic Web Search and Metadata Engine," In *proceedings of the 13th international conference on Information and knowledge management*, 2004, pp. 461-468.

[23] L. Ding, T. Finin, A. Joshi, Y. Peng, R. Pan, and P. Reddivari, "Search on the semantic web," *IEEE Computer society,* Vol. 38, no. 10, Oct. 2005, pp. 62-69.

[24] M. B. Leiner, V. G. Cerf, D. D. Clark, R. E. Kahn, L. Kleinrock, D. C. Lynch, J. Postel, L. G. Roberts, and S. Wolff, "A brief history of the Internet," *ACM SIGCOMM Computer Communication Review,* Vol. 39, no. 5 , 2009, pp. 22-31.

[25] J. Guice, "Looking backward and forward at the Internet," *The Information Society*, Vol.14, no. 3, 1998, pp. 201-211.

[26] M. B. Leiner, V. G. Cerf, D. D. Clark, R. E. Kahn, L. Kleinrock, D. C. Lynch, J. Postel, L. G. Roberts, and S. S. Wolff, "The past and future history of the Internet," *Communications of the ACM,* Vol. 40, no. 2, 1997, pp. 102-108.

[27] P. Atzeni, G. Mecca, and P. Merialdo, "To weave the web," In *VLDB*, vol. 97, 1997, pp. 25-29.

[28] P. Andersen, “What is Web 2.0? ideas, technologies and implications for education,” Vol. 1, No. 1, Bristol, UK, JISC, 2007.

[29] J. desRivieres and J. Wiegand, "Eclipse: A platform for integrating development tools," *IBM Systems Journal*, Vol. 43, No. 2, 2004, pp. 371-383.

[30] S.Brin and L. Page, "The anatomy of a large-scale hypertextual Web search engine," *Computer networks and ISDN systems,* Vol. 30, no. 1, 1998, pp. 107-117.

[31] G. Ozsoyoglu, and A. Al-Hamdani, "Web information resource discovery: Past, present, and future," In *Computer and Information Sciences-ISCIS, 2003*, Springer Berlin Heidelberg, 2003, pp. 9-18.

[32] D. Sullivan, "Major search engines and directories," *Search Engine Watch, 2004*.

[33] M. I. Mauldin, "Lycos: design choices in an Internet search service," *IEEE Expert*, Vol. 12, no.1, Jan 1997, pp. 8-11.

[34] T. Seymour, D. Frantsvog, and S. Kumar, "History of search engines," *International Journal of Management & Information Systems (IJMIS),* Vol. 15, no. 4 , 2011, pp. 47-58.

[35] J. Battelle, "The birth of Google," *WIRED-SAN FRANCISCO*, Vol. 13, no. 8, 2005.

[36] M. Peshave and K. Dezhgosha, "How Search Engines Work: and a Web Crawler Application," PhD dissertation, University of Illinois Springfield, 2005.

[37] A. Heydon and Marc Najork, "Mercator: A scalable, extensible web crawler," *World Wide Web* Vol. 2, no. 4 , 1999, pp. 219-229.

[38] A. Arasu, J. Cho, H. Garcia-Molina, A. Paepcke, and S. Raghavan, "Searching the web," *ACM Transactions on Internet Technology (TOIT),* Vol. 1, no. 1 , 2001 , pp. 2-43.

[39] A. K. Sharma and A. Dixit, "Self adjusting Refresh Time based Architecture for incremental web crawler," *Journal of Computer Science,* Vol. 8, no. 12 , 2008, pp. 349-354.

[40] J. Pokorný, "Web searching and information retrieval," *Computing in Science & Engineering,* Vol. 6, no. 4, 2004, pp. 43-48.

[41] W. B. Croft, D. Metzler, and T. Strohman. *Search engines: Information retrieval in practice*. Addison-Wesley, 2010.

[42] S. Brin, and L. Page, "The PageRank citation ranking: bringing order to the Web," 2006.

[43] J. M. Kleinberg, "Authoritative sources in a hyperlinked environment," *Journal of the ACM* , Vol. 46, no. 5 ,1999, pp. 604-632.

[44] A. Farahat, T. LoFaro, J. C. Miller, G. Rae, and L. A. Ward, "Authority rankings from HITS, PageRank, and SALSA: Existence, uniqueness, and effect of initialization," *SIAM Journal on Scientific Computing* Vol. 27, no. 4 ,2006,pp. 1181-1201.

[45] J. Hendler, "Agents and the semantic web," *IEEE Intelligent systems*, Vol. 16, No. 2, 2001, pp. 30-37.

[46] T. B. Lee, J. Hendler, O. Lassila, "The Semantic Web," *Scientific American Journal,* Vol. 284, 2001, pp. 34-43.

[47] S. Decker, S. Melnik, F. V. Harmelen, D. Fensel, M. Klein, J. Broekstra, M. Erdmann, and I. Horrocks, "The semantic web: The roles of XML and RDF," *IEEE, Internet Computing,* Vol. 4, no. 5 , 2000, pp. 63-73.

[48] V. Dhingra, and K K Bhatia, "SemCrawl: Framework for Crawling Ontology Annotated Web Documents for Intelligent Information Retrieval," In *Intelligent Distributed Computing*, Springer International Publishing, 2015, pp. 213-223.

[49] G. Antoniou and F.V. Harmelen. *A semantic web primer*. MIT press, 2004.

[50] T. Bray, J. Paoli, C. M. Sperberg-McQueen, E. Maler, F. Yergeau and J.Cowan, "Extensible Markup Language (XML)1.1(Second Edition)," W3C Recommendation, 2006.

[51] G. Klyne and J. J. Carroll, "Resource Description Framework (RDF): Concepts and abstract syntax," W3C Recommendation, 2004.

[52] J. J. V. der Ham, F. Dijkstra, F. Travostino, H. M A Andree, and Cees TAM de Laat, "Using RDF to describe networks," *Future Generation Computer Systems,* Vol. 22, Issue 8, 2006, pp. 862-867.

[53] P. Hitzler, M. Krotzsch, and S. Rudolph, *Foundations of semantic web technologies*. CRC Press, 2011.

[54] D. Brickley and R.V Guha, "RDF Vocabulary Description Language 1.0: RDF Schema," W3C Recommendation, February 2004

[55] T.B. Lee, "Semantic web road map", 1998. Available online at http://www.w3.org/DesignIssues/Semantic.html.

[56] J. A. Gerber, A. Barnard, and A. J. Van der Merwe, "Towards a semantic web layered architecture," 2007.

[57] R. Studer, S. Grimm, and A. Abecker, *Semantic web services: concepts, technologies, and applications*. Springer Science & Business Media, 2007.

[58] K. S. Candan , H. Liu, and R. Suvarna, "Resource description framework: metadata and its applications," *ACM SIGKDD Explorations Newsletter,* Vol. 3, no. 1 , 2001, pp. 6-19.

[59] J. Greenberg, "Metadata and the world wide web," *Encyclopedia of library and information science* , Vol. 3, 2003, pp. 1876-1888.

[60] Dublin Core Metadata Editor (DCDOT), 2000. http://www.ukoln.ac.uk/metadata/dcdot.

[61] The Protege Project,2000. http://protege.stanford.edu

[62] H. Knublauch, R. W. Fergerson, N. F. Noy, and M. A. Musen, "The Protégé OWL plugin: An open development environment for semantic web applications," In *The Semantic Web–ISWC 2004*, Springer Berlin Heidelberg, 2004, pp. 229-243.

[63] S-Figueroa, M. Carmen, A. Gómez-Pérez, E. Motta, and A. Gangemi. *Ontology engineering in a networked world*. Springer Science & Business Media, 2012.

[64] P. Haase, H. Lewen, R. Studer, D. T. Tran, M. Erdmann, M. d'Aquin, and E. Motta, "The neon ontology engineering toolkit," *WWW* ,2008.

[65] S. Youn, and Dennis McLeod. "Ontology development tools for ontology-based knowledge management.", 2006.

[66] Apollo Semantic Web Project. http://apollo.open.ac.uk/index.html

[67] J. C., Arpírez, O. Corcho, M. Fernández-López, and A. Gómez-Pérez, "WebODE: a scalable workbench for ontological engineering," In *Proceedings of the 1st international conference on Knowledge capture*, ACM, 2001, pp. 6-13.

[68] N. Guarino, "*Formal ontology in information systems", Proceedings of the first international conference (FOIS'98), June 6-8, Trento, Italy*. Vol. 46. IOS press, 1998.

[69] N. Guarino, Daniel Oberle, and Steffen Staab, "What is an Ontology?," In *Handbook on ontologies*. Springer Berlin Heidelberg, 2009, pp. 1-17.

[70] T. R. Gruber, "A translation approach to portable ontology specifications," *Knowledge acquisition,* Vol. 5, no. 2 , 1993, pp. 199-220.

[71] W. N. Borst, *Construction of engineering ontologies for knowledge sharing and reuse*. Universiteit Twente, 1997.

[72] R. Studer, V. R. Benjamins and D. Fensel, "Knowledge engineering: principles and methods," *Data & knowledge engineering* Vol. 25, no. 1, 1998, pp. 161-197.

[73] L. Yu. *Introduction to the semantic web and semantic web services*. CRC Press, 2007.

[74] K. K. Breitman, M. A. Casanova, and W. Truszkowski, "Ontology in computer science." *Semantic Web: Concepts, Technologies and Applications* ,2007,pp. 17-34.

[75] N. F. Noy, D. L. McGuinness, "Ontology development 101 : A Guide to Creating your first ontology," 2001.

[76] A. Eardley, *Innovative Knowledge Management: Concepts for Organizational Creativity and Collaborative Design: Concepts for Organizational Creativity and Collaborative Design*, IGI Global, 2010.

[77] A. Gómez-Pérez and Oscar Corcho, "Ontology languages for the semantic web," *IEEE Intelligent Systems,* Vol. 17, no. 1 , 2002, pp. 54-60.

[78] P. D. Karp, V. K. Chaudhri, and J. Thomere, "XOL: An XML-based ontology exchange language," 2000.

[79] R. E. Kent, "Conceptual knowledge markup language: the central core,"2011.

[80] J. Heflin, J. A. Hendler, and S. Luke, "SHOE: A Blueprint for the Semantic Web," *Spinning the Semantic Web*, 2003, pp. 1-19.

[81] J. Heflin, J. Hendler, and S. Luke. "Shoe: A prototype language for the semantic web," *Linköping Electronic Articles in Computer and Information Science* , 2001.

[82] J.Heflin, James Hendler, and Sean Luke, "SHOE: A knowledge representation language for internet applications," 1999.

[83] S.Luke and J. Heflin, "SHOE 1.01. Proposed specification," *Shoe Project* ,2000.

[84] *Searching the Web with SHOE*. Defence Technical Information Center, 2000.

[85] D. Brickley and R. V. Guha, "Resource Description Framework (RDF) Schema Specification 1.0: W3C Candidate Recommendation 27 March 2000," 2000.

[86] S. Decker, P. Mitra, and S. Melnik. "Framework for the semantic Web: an RDF tutorial." *IEEE Internet Computing, Vol.* 4, no. 6, 2000, pp. 68-73.

[87] M. A Casanova, W. Truszkowski, and K. Breitman, "Semantic Web: Concepts, Technologies and Applications," *NASA Monographs in Systems and Software, Springer* , 2007.

[88] D. Fensel, F. Van Harmelen, I. Horrocks, D. L. McGuinness, and P. F. Patel-Schneider, "OIL: An ontology infrastructure for the semantic web," *IEEE intelligent systems,* Vol. 16, no. 2 , 2001, pp. 38-45.

[89] D. Fensel, I. Horrocks, F. Van Harmelen, S. Decker, M. Erdmann, and M. Klein. "OIL in a nutshell." In *Knowledge Engineering and Knowledge Management Methods, Models, and Tools*, Springer Berlin Heidelberg, 2000, pp. 1-16.

[90] F. van Harmelen and I. Horrocks, "FAQs on OIL: the ontology inference layer," 2000, pp. 69-72.

[91] I. Horrocks, "DAML+OIL: A Description Logic for the Semantic Web," *IEEE Data Eng. Bull.* 25, no. 1, 2002, pp. 4-9.

[92] L. D. McGuinness and F. V. Harmelen, "OWL web ontology language overview," *W3C recommendation* 10, no. 10, 2004.

[93] M. Dean, G. Schreiber, S. Bechhofer, F. van Harmelen, J. Hendler, I. Horrocks, D. L. McGuinness, P. F. Patel-Schneider, and L. A. Stein, "OWL web ontology language reference," *W3C Recommendation February, Vol.* 10 ,2004.

[94] H. Stuckenschmidt and F. van Harmelen, "Ontology languages for the Semantic Web," *Information Sharing on the Semantic Web*, 2005, pp. 45-61.

[95] R. Mizoguchi and K. Kozaki, "Ontology engineering environments," In *Handbook on ontologies*, Springer Berlin Heidelberg, 2009, pp. 315-336..

[96] J. H. Gennari, M. A. Musen, R. W. Fergerson, W. E. Grosso, M. Crubézy, H. Eriksson, N. F. Noy, and S. W. Tu, "The evolution of Protégé: an environment for knowledge-based systems development," *International Journal of Human-computer studies* Vol. 58, no. 1, 2003, pp. 89-123.

[97] O. Corcho, M. Fernández-López, A. Gómez-Pérez, and Ó. Vicente, "WebODE: An integrated workbench for ontology representation, reasoning, and exchange," In *Knowledge Engineering and Knowledge Management: Ontologies and the Semantic Web*, pp. 138-153. Springer Berlin Heidelberg, 2002.

[98] J. C. Arpírez, O. Corcho, M. Fernández-López, and A. Gómez-Pérez, "WebODE: a scalable workbench for ontological engineering," In *Proceedings*

of the 1st international conference on Knowledge capture, ACM, 2001, pp. 6-13.

[99] A. Kalyanpur, B. Parsia, E. Sirin, B. C. Grau, and J. Hendler, "Swoop: A web ontology editing browser," *Web Semantics: Science, Services and Agents on the World Wide Web* Vol. 4, no. 2 , 2006, pp. 144-153.

[100] A. Kalyanpur, E. Sirin, B. Parsia, and J. Hendler, "Hypermedia inspired ontology engineering environment: Swoop," In *Proceedings of 3rd International Semantic Web Conference (ISWC-2004), Japan* , 2004.

[101] E.Sirin, B. Parsia, B. C. Grau, A. Kalyanpur, and Y. Katz, "Pellet: A practical owl-dl reasoner," *Web Semantics: science, services and agents on the World Wide Web* Vol. 5, no. 2, 2007, pp. 51-53.

[102] P. Haase, H. Lewen, R. Studer, D. T. Tran, M. Erdmann, M. d'Aquin, and E. Motta, "The neon ontology engineering toolkit," *WWW*, 2008.

[103] NeOn toolkit Project. http://www.neon-project.org/.

[104] A. Adamou, R. Palma, P. Haase, E. M. Ponsoda, G. Aguado de Cea, A. Gómez-Pérez, W. Peters, and A. Gangemi, "The NeOn ontology models," In *Ontology Engineering in a Networked World*, Springer Berlin Heidelberg, 2012, pp. 65-90.

[105] WordNet web resource. http//wordnet.princeton.edu/.

[106] H.Boley, S. Tabet, and G. Wagner, "Design Rationale for RuleML: A Markup Language for Semantic Web Rules," In *SWWS*, Vol. 1, pp. 381-401. 2001.

[107] I.Horrocks, P. F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, and M. Dean. "SWRL: A semantic web rule language combining OWL and RuleML," *W3C Member submission* , 2004.

[108] C. Patel, K. Supekar, Y. Lee, and E. K. Park, "OntoKhoj: a semantic web portal for ontology searching, ranking and classification," In *Proceedings of the 5th ACM international workshop on Web information and data management*, 2003, pp. 58-61.

[109] K. Supekar, C. Patel, and Y. Lee, "Characterizing Quality of Knowledge on Semantic Web," In *FLAIRS Conference*, 2004, pp. 472-478.

[110] Dmoz Open directory project home page. http://www.dmoz.org

[111] L. Ding, T. Finin, A. Joshi, R. Pan, R. S. Cost, Y. Peng, P. Reddivari, V. Doshi, and J. Sachs, "Swoogle: a search and metadata engine for the semantic web," In

Proceedings of the thirteenth ACM international conference on Information and knowledge management, 2004, pp. 652-659.

[112] T. Finin, L. Ding, R. Pan, A. Joshi, P. Kolari, A. Java, and Y. Peng, "Swoogle: Searching for knowledge on the Semantic Web," In *Proceedings of the national conference on artificial intelligence*, Vol. 20, no. 4, 2005.

[113] L.Ding, and T. W. Finin, "Boosting semantic web data access using Swoogle," In *AAAI*, vol. 5, 2005, pp. 1604-1605.

[114] Swoogle Search Engine.http://swoogle.umbc.edu/

[115] G. Cheng, W. Ge, Y. Qu, "Falcons: Searching and browsing Entities on the Semantic Web," WWW 2008, China.

[116] Y. Qu, G. Cheng, "Falcon Concept Search: A Practical Search Engine for Web ontologies," IEEE Transactions on System, Man and Cybernetics, Vol No. 41, No 4, July 2011.

[117] Apache Lucene, "A high-performance, full-featured text search engine library," *URL: http://lucene. apache. org* ,2005.

[118] Y. Zhang, W. Vasconcelos, and D. Sleeman, "Ontosearch: An ontology search engine," In *Research and Development in Intelligent Systems XXI*, Springer London, 2005, pp. 58-69.

[119] J. Bailey, F. Bry, T. Furche, S. Schaffert, "Semantic Web Query Languages," *Encyclopedia of Database Systems, Springer*, 2009, pp. 2583-2586.

[120] Z. Zhang and J. A. Miller, "Ontology query languages for the semantic web: A performance evaluation," PhD diss., University of Georgia, 2005.

[121] R. Fikes, P. Hayes, and I. Horrocks, "OWL-QL—a language for deductive query answering on the Semantic Web," *Web semantics: Science, services and agents on the World Wide Web* Vol. 2, no. 1 , 2004, pp. 19-29.

[122] J. Broekstra, and A. Kampman, "Serql: An rdf query and transformation language," *Submitted to the International Semantic Web Conference, ISWC*,2004.

[123] J. Broekstra, and A. Kampman, "SeRQL: a second generation RDF query language," In *Proc. SWAD-Europe Workshop on Semantic Web Storage and Retrieval*, pp. 13-14. 2003.

[124] L. Ding, T. Finin, A. Joshi, Y. Peng, R. Pan, and P. Reddivari. "Search on the semantic web," Department of Computer Science and Electrical engineering,

University of Maryland Baltimore County, Baltimore, Tech Rep. TR CS-05-09, 2005, pp. 62-69.

[125] K.S. Esmaili, and H. Abolhassani, "A Categorization Scheme for Semantic Web Search Engines," In *AICCSA*, 2006, pp. 171-178.

[126] M.Hepp, K. Siorpaes, and D. Bachlechner, "Towards the semantic web in e-tourism: can annotation do the trick?," 14th European Conference on Information System (ECIS), 2006.

[127] U. Shah, T. Finin, A. Joshi, R. S. Cost, and J. Matfield, "Information retrieval on the semantic web," In *Proceedings of the eleventh international conference on Information and knowledge management,* ACM, 2002, , pp. 461-468.

[128] S. L. Tomassen, "Conceptual Ontology Enrichment for Web Information Retrieval," PhD dissertation, Norwegian University of Science and Technology, 2011.

[129] R. Navigli, "Word sense disambiguation: A survey." *ACM Computing Surveys (CSUR)* Vol. 41, no. 2 , 2009.

[130] D. Yarowsky, "Unsupervised word sense disambiguation rivaling supervised methods," *In Proceedings of the 33rd annual meeting on Association for Computational Linguistics,* 1995. pp. 189-196.

[131] D.E.Goldschmidt and M. Krishnamoorthy, "Architecting a search engine for the semantic web*,"* In *Proc. of the AAAI workshop on contexts and ontologies: theory, practice and applications*, 2005.

[132] H. Team, "Semantic Search Technology–making sense of the worlds Information," *White paper Jan* , 2010.

[133] A. Hands, "Duckduckgo http://www. duckduckgo. com or http://www. ddg. gg." *Technical Services Quarterly,* Vol. 29, no. 4, 2012, pp. 345-347.

[134] K. Dahlgren, "Technical overview of Cognition's semantic NLP (as applied to search)," *ReCALL* , 2007.

[135] A. Radhakrishnan, "SenseBot: Summarizing Search Engine results," *Search engine Journal*, 2007. http://www.sensebot.net/news.htm.

[136] T. Converse, R. M. Kaplan, B. Pell, S. Prevost, L. Thione, and C. Walters, "Powerset's Natural Language Wikipedia Search Engine," *Wikipedia and Artificial Intelligence* ,2008.

[137] M.d'Aquin, C.Baldassarre, L.Gridinoc, M.Sabou, S.Angeletou, and E. Motta, "Watson: Supporting next generation semantic web applications," in *Proceedings IADIS International Conference WWW/Internet*, 2007, pp.363-171.

[138] M. Weiten, *Ontostudio as a ontology engineering environment*. Springer Berlin Heidelberg, 2009.

[139] C. Golbreich, "Combining rule and ontology reasoners for the semantic web," In *Rules and rule markup languages for the semantic Web*, Springer Berlin Heidelberg, 2004, pp. 6-22.

[140] E. Sirin and B. Parsia, "SPARQL-DL: SPARQL Query for OWL-DL," In *OWLED*, Vol. 258, 2007.

[141] D.Tsarkov and I. Horrocks, "FaCT++ description logic reasoner: System description," In *Automated reasoning*, Springer Berlin Heidelberg, 2006, pp. 292-297.

[142] V. Dhingra and K. K. Bhatia, "Development of ontology in Laptop Domain for Knowledge Representation," In *International Conference on Information and Communication Technologies, ICICT*, Elsevier Procedia, 2014.

[143] V. Dhingra and K. K. Bhatia, "SemIndex: Efficient indexing mechanism for ontologies," In IEEE *Contemporary Computing (IC3), 2014 Seventh International Conference*, 2014, pp. 340-345.

[144] L. Ding, P.Kolari, Z. Ding, and S. Avancha, "Using ontologies in the semantic web: A survey," In *Ontologies*, Springer US, 2007, pp. 79-113.

[145] DAML Library. http://www.daml.org/ontologies/

[146] Ontolingua Library. http://www-ksl-svc.stanford.edu:5915/

[147] M. Horridge, N. Drummond, J. Goodwin, A. L. Rector, R. Stevens, and H. Wang,"The Manchester OWL Syntax," In *OWLed*, Vol. 216, 2006.

[148] L.Dodds, "Slug: A semantic web crawler," In *Proceedings of Jena User Conference*, 2006.

[149] H. Dong, F. K. Hussain, and E. Chang, "A semantic crawler based on an extended CBR algorithm," In *On the Move to Meaningful Internet Systems: OTM 2008 Workshops*, Springer Berlin Heidelberg, 2008, pp. 1076-1085.

[150] L.Ding, R. Pan, T. Finin, A. Joshi, Y. Peng, and P. Kolari, "Finding and ranking knowledge on the semantic web," In *The Semantic Web–ISWC 2005*, Springer Berlin Heidelberg, 2005, pp. 156-170.

[151] S.K. Staab, K. Apsitis, S. Handschuh, and H. Oppermann, "Specification of an RDF Crawler," 2004.

[152] M.Biddulph, “Crawling the Semantic Web. BBC London,” United Kingdom, 2003.

[153] A. Chakravarthy, "Mining the semantic web," 2005.

[154] V. Jalali, M. Zhou, and Y. Wu, “A study of RDF based data management techniques,” Lecture Notes in computer Science, Vol. 6897, 2011, pp. 366-378.

[155] A. Harth and S. Decker, “Optimized index structures for querying RDF from the web,”In *Proceedings of the LA-Web Congress*, Nov.2005.

[156] D. Beckett, “The design and implementation of Redland RDF application framework,”*Proceedings of 10th International Conference on World Wide Web*, USA, 2001.

[157] S. Harris, N. Gibbins, “3 store: Efficient bulk RDF Store,” In *proceedings of PSSS’03*, 2003, pp. 1-15.

[159] A. Reggiori, “RDFStore Perl API for RDF Storage,” 2002.

[160] D. J. Abadi, A. Marcus, S. Madden, K. Hollenbach, “Scalable semantic web data management using vertical partitioning,” In *VLDB*, 2007, pp. 411-422.

[161] D. J. Abadi, A. Marcus, S. Madden, K. Hollenbach, “SW-Store: a vertically partioned DBMS for semantic web data management”, VLDB J., 2009.

[162] K. Wilkinson, "Jena property table implementation,“Jena Property Table Implementation”, In *SSWS*, 2006.

[163] K. Wilkinson, C. Sayers, H. Kuno, D. Reynolds, “Efficient RDF Storage and Retrieval in Jena”, SWDB 2003.

[164] S. Alexai, V. Christophides, G. Karvounarki, D. Plxousakis, K. Tolle, “RDFSuite: Managing Voluminous RDF Description bases”, 2nd International Workshop on Semantic Web (SemWeb’01), Hong Kong, 2001.

[165] J. Broekstra , A. Kampman , F. Harmelen, “Sesame: A Generic Architecture for Storing and Querying RDF and RDFSchema”, Proceedings of the First International Semantic Web Conference, 2002, pp. 54-68.

[166] S. Harris, N. lamb, N. Shadbol, “4 store: The design and implementation of clustered RDF store,” in SSWS2009: Proceedings of 5th International Workshop on scalable Semantic Web knowledge base Systems, 2009.

[167] T. Finin and Li Ding, "Search engines for semantic web knowledge," *Darpa Agent Markup Language (DAML) tools for supporting intelligent annotation, sharing and retrieval, 2007.*

[168] H. Alani, C. Brewster, and N. Shadbolt, "Ranking ontologies with AKTiveRank," In *The Semantic Web-ISWC 2006*, Springer Berlin Heidelberg, 2006, pp. 1-15.

[169] H. Alani and C. Brewster, "Metrics for ranking ontologies," In *Proceedings of the 4th International EON Workshop and the 15th International World Wide Web conference, Edinburgh, UK.*

[170] M. Sabou, V. Lopez, and E. Motta, "Ontology selection for the real semantic Web: How to cover the queen's birthday dinner?," In *Managing Knowledge in a World of Networks*, Springer Berlin Heidelberg, 2006, pp. 96-111.

[171] T. Composer, "TOPBRAID COMPOSER 2007 Features and getting Started Guide Version 1.0", created by TopQuadrant, US, 2007.

[172] S.Tartir and I. B. Arpinar ,"Ontology Evaluation and Ranking using OntoQA," in Proceedings of *1st IEEE International Conference on Semantic Computing*, USA, Sep. 2007, pp. 185-192.

[173] Y. Lei , V. Uren and E. Motta, "SemSearch: A search engine for the semantic web," in *Proceeding of the 15th International Conference on MANAGING Knowledge in the World of Network (EKAW)*, Podebrady, Czech Republic, 2006, pp: 238-245.

[174] A. Hogan, A. Harth and S. Decker, "ReConRank: A Scalable Ranking Method for Semantic Web Data with Context," In *Proceedings of the 2nd Int. Workshop on Scalable Semantic Web Knowledge Base Systems (SSWS 2006),* The 5th Int. Semantic Web Conf., Athens, Georgia, USA,2006.

[175] P. Buitelaar, T. Eigner, & T. Declerck, "*OntoSelect: A Dynamic Ontology Library with Support for Ontology Selection*," Proceedings of the Demo Session at the *International Semantic Web Conference*. Hiroshima, Japan, 2004.

[176] S. Bechhofer, I. Horrocks, C. Goble, and R. Stevens, "OilEd: a reason-able ontology editor for the semantic web," In *KI 2001: Advances in Artificial Intelligence*, pp. 396-408, Springer Berlin Heidelberg, 2001.

[177] Y. Sure, J. Angele, and S.Staab, "OntoEdit: Guiding ontology development by methodology and inferencing," In *On the Move to Meaningful Internet Systems,* Springer Berlin Heidelberg, 2002, pp. 1205-1222.

[178] L. Miller, A. Seaborne, and A. Reggiori, "Three implementations of SquishQL, a simple RDF query language," In *The Semantic Web—ISWC 2002*, Springer Berlin Heidelberg, 2002, pp. 423-435.

[179] A. Seaborne, "Rdql-a query language for rdf," *W3C Member submission*, 2004.

[180] M. J. O 'Connor, and A. K. Das, "SQWRL: A Query Language for OWL," in *OWLED*, Vol. 529, 2009.

[181] Y. Zhang, W. Vasconcelos, and D. Sleeman, "Ontosearch: An ontology search engine," In *Research and Development in Intelligent Systems XXI*, Springer London, 2005, pp. 58-69.