ONTOLOGY BASED INFORMATION RETRIEVAL SYSTEM FOR HIDDEN WEB

THESIS

submitted in fulfillment of the requirement of the degree of

DOCTOR OF PHILOSOPHY

to

YMCA UNIVERSITY OF SCIENCE & TECHNOLOGY

by

MANVI

Registration No: YMCAUST/Ph15/2010

Under the Supervision of

Dr. KOMAL KUMAR BHATIA
PROFESSOR

Dr. ASHUTOSH DIXIT
ASSOCIATE PROFESSOR

Department of Computer Engineering
Faculty of Engineering and Technology
YMCA University of Science &Technology
Sector-6, Mathura Road, Faridabad, Haryana, INDIA

February, 2017

DECLARATION

I hereby declare that this thesis entitled "**ONTOLOGY BASED INFORMATION RETRIEVAL SYSTEM FOR HIDDEN WEB"** by **MANVI,** being submitted in fulfilment of the requirements for the Degree of Doctor of Philosophy in Department of Computer Engineering under Faculty of Engineering and Technology of YMCA University of Science & Technology, Faridabad, during the academic year 2016-2017, is a bona fide record of my original work carried out under the guidance and supervision of **Dr. KOMAL KUMAR BHATIA, PROFESSOR** and **Dr. ASHUTOSH DIXIT, ASSOCIATE PROFESSOR, DEPARTMENT OF COMPUTER ENGINEERING, YMCA UNIVERSITY OF SCIENCE & TECHNOLOGY, FARIDABAD** 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.

(MANVI)

Registration No: YMCAUST/Ph15/2010

CERTIFICATE

This is to certify that this thesis entitled **“ONTOLOGY BASED INFORMATION RETRIEVAL SYSTEM FOR HIDDEN WEB”** by **MANVI** submitted in fulfillment of the requirements for the award of Degree of Doctor of Philosophy in Department of Computer Engineering, under Faculty of Engineering and Technology of YMCA University of Science and Technology, Faridabad, during the academic year 2016-17, is a bona fide record of work carried out under our guidance and supervision.

We further declare that to the best of our 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.

Dr. KOMAL KUMAR BHATIA
Professor
Department of Computer Engineering
Faculty of Engineering and Technology
YMCA University of Science & Technology, Faridabad

Dr. ASHUTOSH DIXIT
Associate Professor
Department of Computer Engineering
Faculty of Engineering and Technology
YMCA University of Science & Technology, Faridabad

Dated:

ACKNOWLEDGEMENT

I express my gratitude to almighty God for giving me strength and courage to complete this thesis.

I would like to express my sincere and deep gratitude to my *Gurus* **Dr. Komal Kumar Bhatia**, Professor, Department of Computer Engineering and **Dr. Ashutosh Dixit,** Associate Professor, Department of Computer Engineering, YMCA University of Science & Technology, Faridabad for giving me the opportunity to work in this area. It would never be possible for me to take this thesis to this level without their innovative ideas, invaluable guidance, continuous support and encouragement. Their knowledge of different perspectives of research provided me with the opportunity to broaden my knowledge and to make significant progress. At times when was I stumbled upon big obstacles, both my mentors encouraged me to look further and keep sailing through tough times.

I gratefully acknowledge my university colleagues for their encouragement, support and invaluable suggestions in completing this research. I am also thankful to my students who helped me directly and indirectly in completing my research work.

Words fail to express my heartfelt thanks to my better half **Mr. Pramesh Dahiya** for his technological assistance and unparalled availability at all times during the course of my work. I would also like to thank my mother **Mrs. Santosh Siwach** and sister **Ms. Shashi Siwach** for being a constant source of support and motivation which help me to reach here.

My special thanks to my lovely daughter, **Vasundhara** for understanding me and giving me time for doing my research work. A word of special thanks is due to my niece **Ms. Ankita Gill** for baby sitting my three months old lovely son **Maanvendra,** so that i can complete the writingesis. I would also like to express my thanks to my YMCAUST friends for being there always in my tough times.
Thank you all**!**

(Manvi)

ABSTRACT

Hidden web data is the huge and high quality data which is hidden behind form interfaces. This data is very dynamic and is increasing in volume tremendously. The simple crawler which downloads web pages by following hyperlink structure is not able to extract these hidden web contents. For extracting this data special crawlers are designed which works on two basis. The first type of crawlers download all the form interfaces integrate them and produce a new interface where a user enters query. The query terms then are submitted to form interfaces individually and corresponding pages are downloaded. The second type of crawlers does not integrate the form interfaces. After getting the query terms by user, these crawlers fill in each interface with the terms and extract the hidden web data.

Both types of crawlers traditionally use simple label value technique to find the values that are required to fill in various fields of a form interface. These crawlers maintain a simple table as database to store the values corresponding to different labels. This scheme is not an efficient method as crawler has a finite set of values to choose from. Moreover basic hidden web crawlers do not have any process to decide which values from the domain are semantically correct.

Hence there is a requirement of designing an information retrieval system which is so intelligent that it chooses the exact and correct values to be filled on the interfaces. This system crawler can be made intelligent by attaching meaning to the data. The semantics or meaning to data can be attached in form of relations with other terms with the help of ontology. This ontology can also be used during indexing of hidden web data retrieved and downloaded by the system. To present the data to users efficiently a ranking technique is also needed to be developed.

Ontology is a way to define any entity with help of relationship this entity holds with other entities. Ontology defines any entity in form of classes and attaches meaning to the data by attaching the relationship with other classes. By involving ontology for extracting hidden web data the exact values for various fields in form interfaces are found. The work is targeted to design an information retrieval system based upon ontology to extract hidden web data.

The thrust of the thesis is to extract hidden web data using ontology and present it to the end user efficiently. For this, first of all a domain specific ontology is constructed using Protégé

for book domain. Protégé is an open source tool available to construct ontology developed by Stanford. The book domain ontology developed serve as prototype for further generating domain specific ontology from web pages automatically.

A novel technique for generating ontology automatically has been proposed and implemented. This ontology solves the purpose of creating a semantic database. The semantic database is created in oracle in form of Subject, Predicate and Object triples. This database is used to store various instance values from ontology. These values are used by the proposed ontology based hidden web crawler (OHWC) to generate queries for hidden web interfaces. The queries thus generated are having the exact match of values corresponding to each field of interface. The above said ontology is also used in mapping process of hidden web crawler designed in this thesis.

The work has also put forward a solution to the inefficiency in ontology mapping by proposing a new method which removes the irrelevant pairing of nodes. The OHWC downloads the hidden web pages by mapping the form interface with the developed domain specific ontology. The implementation of this crawler offered improved results as compared to previous hidden web crawlers.

The next module of the thesis is motivated by the fact that petite work has been done towards indexing of hidden web data. The indexing schemes developed so far are not developed keeping hidden web in mind. These techniques are also not able to resolve the ambiguity problem in terms. Hence an ontology based indexing technique is proposed which not only attaches the concept with indexed term but also saves the context in which the term is used.

During the course of this research, the need of a new ranking technique identified. In this context the thesis provides a novel ranking approach for hidden web data. The ranking method proposed here uses both query independent and query dependent factors to calculate the final rank score.

The multifold contributions of the thesis include: i*) Automatic generation of domain specific ontology ii) Design and implementation of ontology based hidden web crawler iii) Proposed novel ontology based indexing technique for hidden web and iv) A new ranking technique for hidden web data.*

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

WWW:	World Wide Web
HTTP:	Hypertext Transfer Protocol
URL:	Uniform Resource Locator
BFS:	Breadth First Search
DFS:	Depth First Search
HITS:	Hyperlink Induced Topic Search
PR:	Page Rank
LVS:	Label Value Set
XML:	Mark-up Language
PIW:	Publically Indexable web
IR:	Information Retrieval
D:	Domain
DNS:	Domain Name Resolver
URL:	Uniform Resource Locator
URI:	Uniform Resource Identifier
FTP:	File Transfer Protocol
IP:	Internet Protocol
DUST:	Different URLs Same Text
PARCAHYD:	Parallel Crawler using Augmented Hypertext Documents
SIG:	Signature
DOM:	Document Object Model
HTML:	Hyper Text Mark-up Language
HiWE:	Hidden Web Extractor
XHTML:	Extensible Hyper Text Mark-up Language

DTD:	Document Type Definition
AHP:	Analytical Hierarchy Process
OHWC:	Ontology based Hidden Web Crawler
RDF:	Resource Description Framework
OWL:	Ontology Web Language
SPO:	Subject,Predicate,Object
N/W:	Network
DSIM:	Domain-Specific Interface Mapper
LOC:	Location
LDB:	Local Database
DOC:	Document
OC:	Occurrence Count
PDF	Portable Document Format
DB	Database
SQL	Structured Query Language

Chapter 1

INTRODUCTION

1.1 GENERAL

Crawlers [1] are the programs that browse the World Wide Web (WWW) in a methodical and automated manner. They automatically traverse the web graph retrieving pages and building a local repository of web they visit. Current day Crawlers as shown in figure 1.1, have only targeted a portion of web called publicly index able web (PIW) [8]. PIW is the set of web pages reachable by crawlers by following hyperlinks; ignoring search forms and pages that require authentication. In particular they ignore the tremendous amount of high quality information hidden beneath search forms and electronic databases.

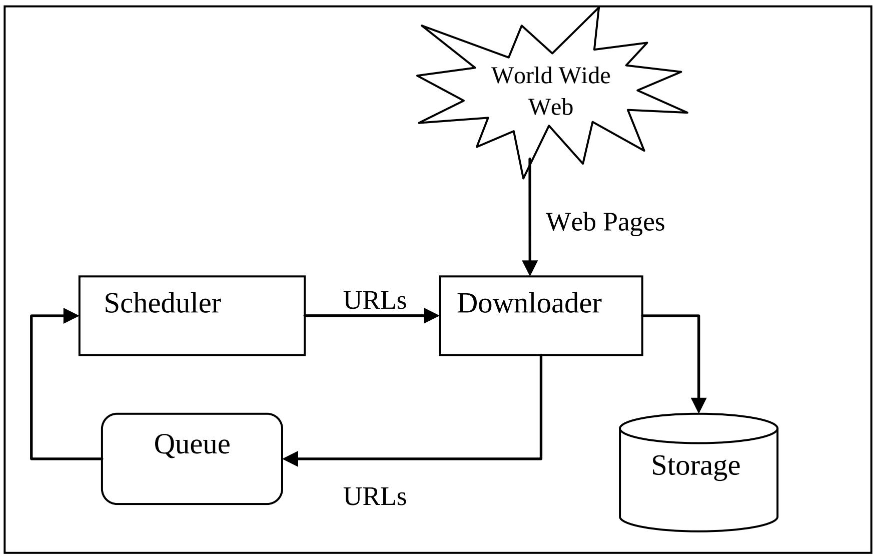


Figure 1.1 Basic Architecture of a Crawler

1.2 MOTIVATION

An ever increasing amount of information on the Web today is available only through search interfaces; the users have to type in a set of keywords in search forms in order to access the pages from certain websites. These pages are often referred to as *Hidden Web.* Recent studies [8,9,10] have estimated the size of this hidden web at around 500 times the size of *PIW*. Simple crawlers are not capable of discovering and indexing these pages because of the following reasons:

i) There are no static links to hidden web pages.

ii) The large amount of high quality information is buried under dynamically generated sites as depicted by figure 1.2.

iii) The only entry point to hidden web site is a query interface.

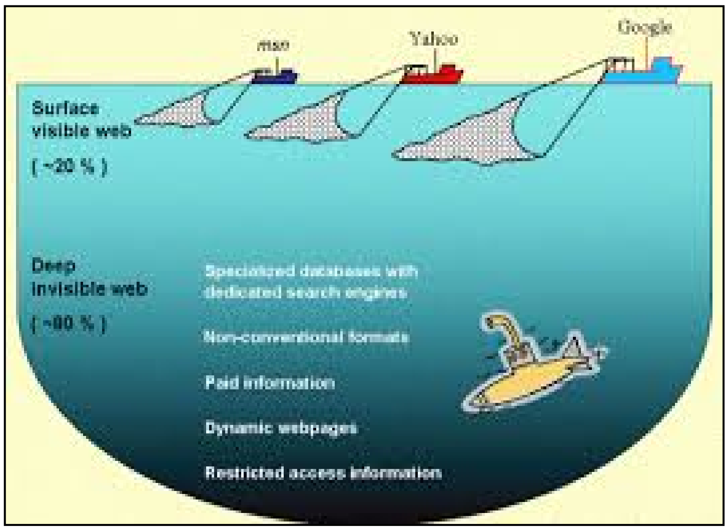
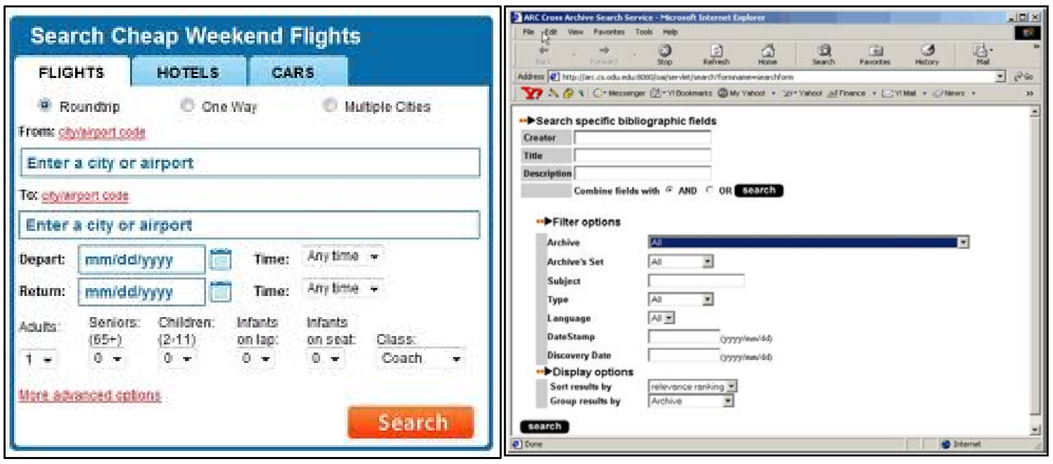


Figure 1.2 Diagram showing % of Hidden Web Pages

Sources/Examples of the Hidden Web can be Publication databases, Library catalogues, Yellow Pages, Books sites, ecommerce sites, airline and railway reservation etc and other directories, Weather services, Geolocalization services, US Census Bureau data etc. as shown in figure 1.3.



Search Cheap Weekend Flights

FLIGHTS | HOTELS | CARS

[✓ Roundtrip] One Way Multiple Cities

From: city/airport code

Enter a city or airport

To: city/airport code

Enter a city or airport

Depart: mm/dd/yyyy Time: Any time

Return: mm/dd/yyyy Time: Any time

Adults: 1; Seniors: (65+) 0; Children: (2-11) 0; Infants on lap: 0; Infants on seat: 0; Class: Coach

More advanced options

Search

ARC Cross Archive Search Service - Microsoft Internet Explorer

Address: http://arc.cs.odu.edu:8080/oai/servlet/search?formname=searchform

Search specific bibliographic fields

Creator

Title

Description

Combine fields with [✓ AND] OR search

Filter options

Archive: All

Archive's Set: All

Subject

Type: All

Language: All

DateStamp (yyyy/mm/dd)

Discovery Date (yyyy/mm/dd)

Display options

Sort results by: relevance ranking

Group results by: Archive

search

Figure 1.3 Examples of Hidden Web Page Interfaces.

For developing a hidden web information retrieval system, following are the main challenges that exist:

i) *Discovery:* Finding the search interfaces that contain hidden web information behind them i.e. discovering the appropriate entry point for the hidden web.

ii) *Analysis:* Analyzing those interfaces to find out which method should be used for filling these interfaces e.g. these interfaces should be merged to develop a new interface or each interface is filled one by one to get the desired result.

iii) *Filling values in the form/interface:* Finding the exact method/algorithm/process to fill those interfaces with relevant values.

iv) *Generating queries:* After finding the exact algorithm for filling form interfaces; there is a need of constructing the queries with appropriate values to get the desired result.

v) *Indexing:* After getting the resultant downloaded pages they should be indexed properly for efficient retrieval.

vi) *Ranking:* For providing relevant results to the user there should be an efficient ranking mechanism that will return result according to the rank assigned.

1.3 EXPLORING THE NEED OF ONTOLOGY IN HIDDEN WEB

The objective of the proposed research work is to develop an information retrieval system where user type in queries and get the desired result from downloaded hidden web content more effectively and efficiently. The proposed work will do ontology based data extraction and information integration.

Ontology provides a common vocabulary of an area and defines, with different level of formality, the meaning of terms and relationships between them [15].

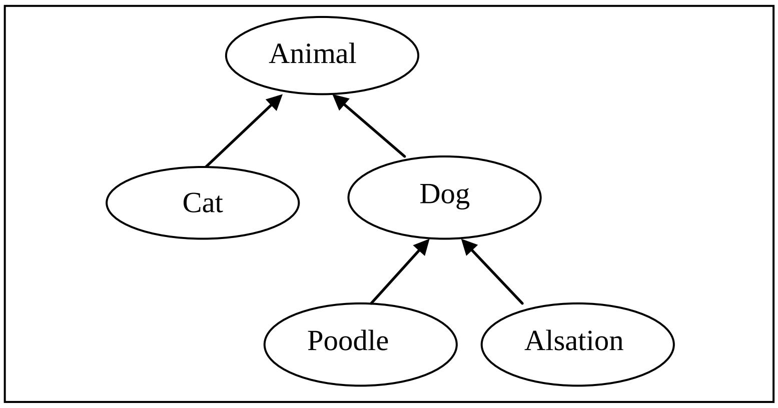


Figure 1.4 Example Showing Animal Ontology having Alsation as Instance of Dog Class.

Use of ontology increases the relevance by involving the relationship and context in the search. Earlier hidden web crawlers [12, 13] use attribute based matching processes to fill the search interface. This work can be done more precisely and efficiently by using ontology which is a formal, explicit specification of a shared conceptualization [11].

Conceptualization refers to an abstract model of phenomena in the world by having identified the relevant concepts of those phenomena. *Explicit* means that the type of concepts used and the constraints on their use are explicitly defined. *Formal* refers to the fact that the ontology should be machine-readable. *Shared* reflects that an ontology should capture consensual knowledge accepted by different communities.

By combining the hidden web retrieval problem with domain specific ontology, the proposed work automatically fills in the text boxes with values from predefined ontology. This not only made the retrieval process task specific, but also increased the likelihood of being able to extract just the relevant subset of data.

1.4 PROBLEMS IN EXISITING LITERATURE

A critical look at the available literature [8, 9, 17, 18, 65, 78] indicates that the following issues need to be addressed towards building an effective information retrieval model for Hidden Web.

i) Relevance: Hidden Web crawlers developed till now are based upon label value matching scheme which is simple keyword searching for retrieving data. This is not an effective way because it does not involve context of the data and making it difficult to obtain good results with high precision.

ii) Database Selection and Implementation: For finding appropriate values which are required to be filled in form interface, a domain specific semantic database is required to be created. The available literature [20] indicates how the hidden web data is collected to create databases but not on how that particular databases are stored so that it may satisfy users queries more suitably.

iii) Automatic and Rich Ontology: Hidden Web services which are mostly form-based interfaces need to be described using new approach that involves attaching meaning to the data. No technique for automatic generation of Ontology with the help of web pages is developed till now.

iv) Efficiency in Mapping: Lack of an effective mapping approach due to pairing of irrelevant nodes during ontology mapping [64, 66]. This issue needs to be addressed towards the efficient design of Ontology Mapping System also considering the proper utilization of resources (mainly in terms of time).

v) Synchronization due to non integration of various phases for developing a whole information retrieval system for hidden web i.e. till now no frame work has been designed which coordinates all the processes like form filling, submitting, downloading the web pages, indexing them and presenting to user that are needed for a system.

vi) Ambiguity in Indexing*:* The work done so far tried to make indexing scheme based upon semantics but none was able to resolve ambiguities in words.

vii) Ranking: There are no proper algorithm defined till now for Indexing and ranking of hidden web pages.

1.5 OBJECTIVES

As the quantity and quality of hidden web is increasing, it becomes more difficult to extract it. This research has focused on following techniques of its retrieval.

1. **To generate Ontology for specific Domain automatically:** For generating and responding to user query, a database that is populated knowledge base which satisfies the user's need of information is needed. Ontology is one of the best ways for creating such knowledge base.

   It has been observed that various researchers have proposed algorithms to access deep web using various techniques for automating the task of extraction of form pages and crawling form pages but none has described the way to automatically find the values that need to be filled in the form page. Also no method till now has been devised to create and populate the values in a database so that more and more values are available to fill various forms.

Solution: *In order to find appropriate values that are needed to fill the form interfaces of hidden web; book domain ontology has been constructed in Protégé initially as a model. Book Domain contains the collection of books that are related to Author, Publication, Award and many more. For proposing the idea four sub domain has been chosen to generate book domain ontology that are Book, Author, Award and Publication. Protégé is used to create the ontology as it is free, open source ontology editor and knowledge-base framework.*

After this a database using ontology has been constructed which is contextual or semantically rich, which is used to fill various label values in a search form. With the help of RDF [50] attached with web pages, ontology is created and stored in database. Thus accurate and meaningful information is retrieved and used for filling up web interfaces.

2. **To Design Ontology based Hidden Web Crawler**: In order to download the Hidden Web contents from the WWW the crawler needs a mechanism for Search Interface interaction i.e. it should be able to download the search interfaces, automatically fill them and submit them to get the Hidden Web pages. Many researchers are trying to develop novel ideas to access hidden web in order to improve searching experience for users.

   Previous hidden web crawlers [17, 18, 19] uses attribute based matching processes to fill the search interface. This work can be done more precisely and efficiently by using ontology.

   Solution: In order to resolve the identified problems Ontology based data extraction and information integration need to be developed. By combining the hidden web retrieval with domain specific ontology, the proposed work automatically fills in the text boxes with values from ontology. This will make not only the retrieval process task specific, but will increase the likelihood of being able to extract just the relevant subset of data. Hence a novel design of ontology based hidden web crawler is proposed and implemented.

3. **To Design mapping technique to map the Domain Specific Ontology with form interface:** Hidden web is accessible only after submitting the queries on

form interfaces. Prior to this; exact values must be filled-in to various fields of query interface, which in turn requires matching the fields present on the form interface to the concepts from the predefined Ontology database of the same domain**.**

Existing ontology mapping system has the improper identification of matching concept pair as well as the lack of an efficient mapping approach when matching between two ontologies.

Solution: *In proposed work ontology mapping system, primary focus is on making the process more efficient in terms of resources consumed during the process. The complexity of matching (in a pair-wise set up) is usually proportional to the size of the ontology under consideration. A straightforward approach is used to reduce the number of pair-wise comparisons. Out of so many possible concept-pairs only a few are considered for mapping based on the depth of the query-form ontology. These concept-pairs are further reduced to a minimal set by eliminating irrelevant pairs.*

4. **To find ontology based indexing technique for hidden web data:** Indexing is performed on the web pages after they have been gathered into a repository by the crawler. The existing architecture of search engine shows that the index is built on the basis of the terms of the document and consists of an array of the posting lists where each posting list is associated with a term and contains the term as well as the identifiers of the documents containing the term.

   The current information retrieval systems use terms to describe documents and search engines consider term frequency as a factor to determine its importance in a page and does not take into account the context(sense) in which term appears in document and how much a term represent a page, which can be done by analysing its relation with other terns by using concept of ontology**.** Most current search engines [76, 77] use indexes that are built at the syntactical level and return hits based on simple string comparisons.

   Solution: *The context of a word to be indexed is compared to the context of Concept in ontology. For each individual word, a mapping score ms(w,c) that* *indicates how good the word w is mapped to ontological concept has been calculated. Each word is replaced by the best matching concept. So all information like synonyms, relations with other words and context in which word is used is available due to use of ontology.*

5. To find a novel ranking technique for hidden web data: Search engines use two different kinds of ranking factors: query-dependent factors and query Independent Factors. Query-dependent factors are specific to a given query are measures such as word documents frequency, the position of the query terms within the result page or the inverted document frequency etc. Query independent factors are attached to the results, regardless of a given query like Link popularity, Click popularity and upto-dateness etc. The PageRank algorithms that are commonly used by the conventional search engines are not effective for Hidden Web pages.

Solution: *The proposed algorithm for ranking uses both factors of query dependent such as page frequency , query – page content matching are used and factors which are independent such as page content popularity , page source rank, user feedback are to design a novel and efficient ranking technique.*

1.6 ORGANIZATION OF THESIS

The thesis has been organized in the following chapters:

Chapter 2: This chapter provides the details about the background study carried out to pursue this research work. A detailed review of selected Hidden Web Crawlers developed so far has been provided. Comparison of different techniques is also been provided in a table. The concept of ontology is introduced here and the exploration of ontology usage for extracting hidden web information has been discussed. This chapter also throws light on various indexing and ranking techniques along with table of comparisons.

Chapter 3: In this chapter a novel architecture of an ontology based information retrieval system for hidden web has been designed and proposed. The architecture depicts different functional modules that are proposed and discussed in next chapters. The basic flow of the system has also been discussed in detail in this chapter

Chapter 4: The first module that is generation of ontology is described in this chapter in detail. The ontology has been designed and generated using Protégé[49] for Book Domain by taking instance values of faculty of YMCA and their publications. The basics of class, properties and attributes has been described and explained in detail. <S,P,O> triples has been defined for book domain.

Chapter 5: This chapter proposes a novel technique for automatic generation of ontology from web pages. The architecture of this module along with procedure to find subject, object and predicate from the source code of web pages is described here. This ontology is more generic as compared to the Ontology generated manually using Protege. Hence both ontologies are merged and a database of <S,P,O> has been designed and created here.

Chapter 6: This Chapter proposes a novel architecture of ontology based hidden web crawler. The details of various modules of this crawler and basic flow with algorithms have been discussed. This crawler is used to extract the hidden web information form the web using ontology. For making the hidden web crawler work, the form interfaces are mapped to predefined ontology to get the exact values to fill various fields on the interface. This chapter also proposes a novel technique for mapping two ontologies.

Chapter 7: In this chapter a new ontology based indexing technique for hidden web contents is proposed. This resolves the problem of ambiguities in terms by attaching both the concept and context of words with each term. This indexing scheme provides an efficient and fast access to the end user. To provide user with a list of relevant pages as a result of query entered a new ranking technique for hidden web data is also proposed and developed here in this chapter.

Chapter 8: This chapter gives a conclusion and provides guidelines for future work in this area.

In Appendix the implementation and snapshots of two domains i.e. Book and Airlines are discussed.

Finally, the bibliography includes references to publications in this area.

Chapter 2

HIDDEN WEB RETRIEVAL AND ONTOLOGY: A REVIEW

2.1 WWW (WORLD WIDE WEB)

Today is the world of information and for getting information user rely upon World Wide Web. The World Wide Web is a global information medium of interlinked hypertext documents accessed via computers connected to the internet. It allows people to share information globally. Internet and WWW are terms used side by side but they are quite different, the web is a service that operates over the Internet. Internet came first then the WWW was developed. WWW allows anyone to read, write and publish documents freely hiding all the details of which protocol is used, exactly where the machine is located geographically, which operating system and software's are installed etc. [2, 22]. It allows users to point to any other Web pages without any restrictions.

WWW is generally divided into two parts: Surface Web and Hidden Web. The Surface Web consists of billions of browsable pages, while the Hidden Web contains hundreds of thousands of data sources. The surface Web refers to the part of the Web that can be crawled and indexed by general purpose search engines, while the hidden Web refers to the abundant information that is hidden behind the query interfaces and not directly accessible to the search engines.

2.1.1 History of WWW

Brin defines the Web as follows (1998) *"The Web is a vast collection of completely uncontrolled heterogeneous documents"* [23]. When a user requests for a web page the page's HTML text is first requested and parsed by the browser, if page contains some pictures of graphics then browser makes additional requests. The Web browser then understands and shows the page to the user as described by the HTML, CSS and incorporating the images and other resources as necessary.

The World Wide Web (WWW) is a huge network of distributed system which contain millions of clients and servers maintained at same or distant geographical positions.

Servers maintain collections of documents, while clients are there to provide access of these documents easily to end user without worrying about where the information or document is present [24]. The Web started as a project at CERN, the European Particle Physics Laboratory in Geneva [25]. It was developed to let its large and geographically dispersed group of researchers provide access to shared documents using a simple hypertext system.

2.1.2 Basics of WWW

The WWW is basically a client-server system with millions of servers distributed around the world. Each server has huge collection of documents and each document whether text, pdf, media etc. is stored in form of file. A client interacts with Web servers through a special application known as a browser. A browser is responsible for displaying a document in exact format as given by server. The Client Server architecture is described in figure 2.1 below.

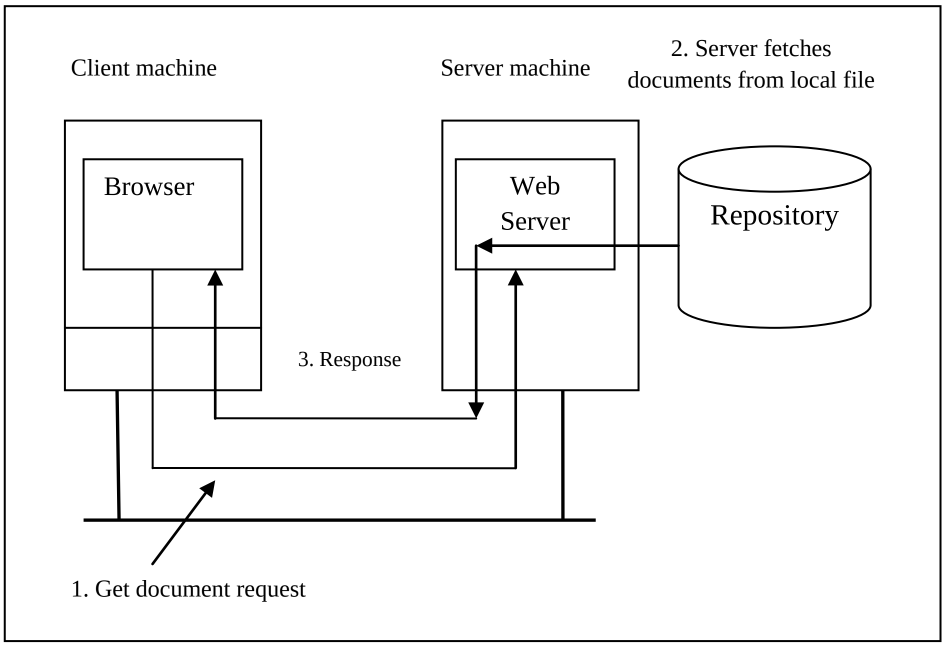


Figure 2.1 Overview of WWW

When a user searches for a document it enters the request through browser to the client requests for fetching a document. The client further sends the request to the corresponding server. The Server then sends a copy of the document back to the client. Also the server adds more documents to its repository coming from clients as a request for storage.

The document or a file is refereed by means of a reference called a Uniform Resource Locator (URL). URL specifies the name of the document along with where a document is located.

2.2 IR (INFORMATION RETREIVAL)

Information retrieval (IR) is an activity by which user can obtain information resources that are relevant to his/her need. In terms of WWW or Internet, it is finding material document of structured/unstructured nature that satisfies an information need from within large collections (usually stored on computers) [4, 6, 7].

Initially information retrieval was use in index searching for example searching authors, title and subjects in library card catalogs or computers. Now a days IR has emerged in new techniques which includes modelling, user interfaces, data visualization, classification and categorization of documents and filtering of data. IR is used these days to represent data, organize the information and also in storage data/information. The most important task information retrieval is dealing with is information extraction i.e. when a user wants some information what techniques should be applied so that he/she retrieve accurate that information easily.

2.2.1 Types of IR

IR can be divided in to two retrieval techniques, one is information retrieval and the other is data retrieval[28]. In data retrieval, the result corresponding to a query entered by user is always precise returning the exactly match records corresponding to a given query. However in information retrieval result can differ along with mild errors because information retrieval deals with natural language documents. These types of documents do not have proper structure and one query may relate to more than one tuple in database.

The data used in Data retrieval is always structured and not ambiguous. The data retrieval cannot provide a solution given a subject or topic, but information retrieval is able to do so. In order to satisfy the user's information needs, the IR system must find a way to interpret the contents of the information items and be able to rank them according to a degree of relevance to the user query [22]. This interpretation involves how to extract information in syntactic and semantic ways.

2.2.2 Goal of IR

The goal of an IR system is to retrieve all the documents, which are relevant to a query while retrieving as few non-relevant documents as possible[29]. To achieve this goal, IR needs users to provide a set of words which convey the semantics of the information need. Also, a document is represented by a set of keywords or index terms for IR to extract. These keywords or index terms are extracted by i) eliminating stop words like articles and connectives, ii) the use of stemming (which reduces distinct words to their common grammatical root), and iii) identifying nouns (which eliminates adjectives, adverbs, and verbs).

2.3 SEARCH ENGINE

WWW is huge and to deal with it is a challenging task. Several studies have estimated the size of the Web, and while they report slightly different numbers, most of them agree that over a billion pages are available. New data and information is uploaded daily on WWW. Today in world of social media tremendous information is flowing at WWW. Keeping aside the newly created pages, the existing pages are continuously updated. For example, in a study of over half a million pages over 4 months, it was found that about 23% of pages changed daily[2, 16]. In the .com domain 40% of the pages changed daily, and the half-life of pages is about 10 days (in 10 days half of the pages are gone, i.e., their URLs are no longer valid).

The World Wide Web allows people to share information globally. The amount of information grows without bound. In order to extract information that user is interested in, a tool to search the Web is needed. This tool is called a *search engine*. The users search documents by keywords. AltaVista, Excite, Google and Northern Light are few examples of search engines are. However, there are also other type of search engines that are specialized in other languages such as Chinese, Korean, and Japanese.

There are differences in the ways various search engines work, but they all perform three basic tasks:

- They search the Internet based on important words.
- They keep an index of the words they find, and where they find them.
- They allow users to look for words or combinations of words found in that index.

2.3.1 Search Engine Architecture

A search engine is an information retrieval system that has been developed to help the user to find information on WWW easily, accurately and quickly. As WWW is vast Web of hyperlinked documents search engines try to help to reduce the time and efforts required to find information. Search engines provide a user interface that enables the users to specify what user wants in form of a query. The query entered by user is typically expressed as a set of words that identify the desired information. Some text search engines require users to enter two or three words separated by white spaces for the search of required information contained in text documents, pictures files, sounds files etc.

The architecture of conventional search engines roughly consists of three components [3] - a crawler, an indexer, and a searcher as shown in figure 2.2. The crawler starts with a set of seed URLs and repeatedly downloads pages, extracts hyperlinks from the pages, and crawls the extracted links. Every engine relies on a crawler module to provide the grist for its operation.

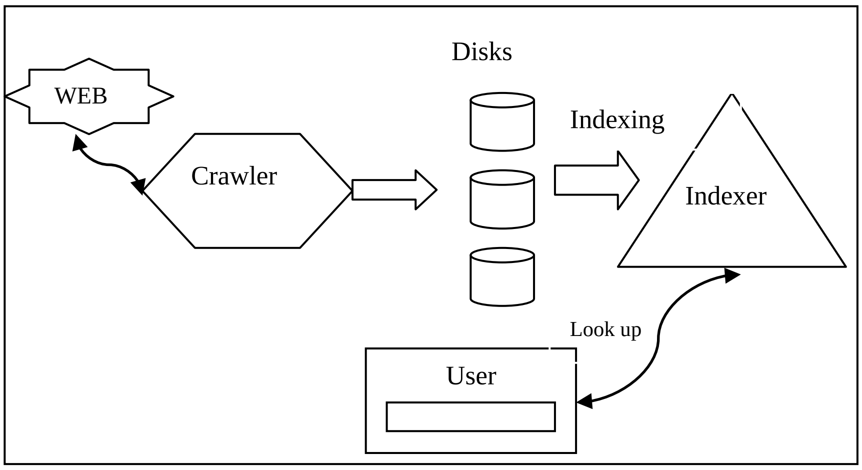


Figure 2.2 Basic Architecture of Search Engine

Crawlers are small programs that 'browse' the Web on the search engine's behalf, similarly to how a human user would follow links to reach different pages. The programs are given a starting set of URLs, whose pages they retrieve from the Web. The indexer performs an indexing function. It reads the repository, uncompresses the documents, and parses them. Each page is converted into a set of word occurrences called hits. The hits contain information about a word: position in document, an approximation of font size, and capitalization.

The detailed architecture of search engine is shown in figure 2.3, which contains the following modules:

i) The publicly available search engine: This includes the HTML interface where the users submit their queries, and the mechanism for serving these queries. This part is very important for the search engines, since this is the only visible part to the end-users. This part is actually more complex than it sounds, but it is completely out of the scope of this work to elaborate on it.

ii) The database: The database stores all the crawled data from the web crawlers. The search engine queries the database in order to answer to any user's request. Furthermore, the database feeds the downloader with URLs to download. The information stored in the database is usually updated from the processor.

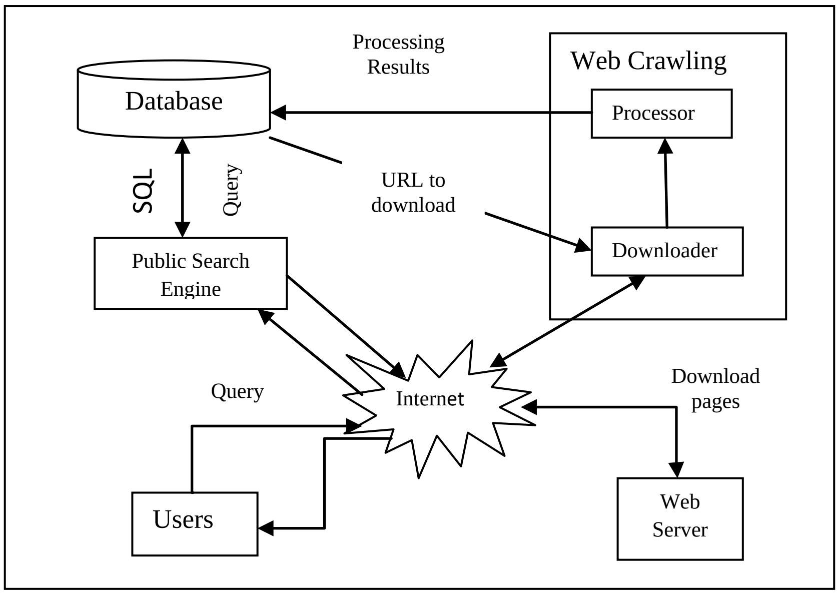


Figure 2.3 Detailed Module Description of a Search Engine.

iii) The web crawling system: The web crawling system is the subsystem responsible for maintaining the search engine database and incorporating the changes from the web. Most of the times multiple instances of this component run in parallel to reduce network bandwidth. The web crawlers download and process as many web pages from the web as possible, by using the standard HTTP protocol.

iv) The query engine: This module is responsible for receiving and fulfilling search requests from users. The engine relies heavily on the indexes, and sometimes on the page repository. Because the size of the web is very large and the user enters one or two keywords which returns a very large set of results.

2.4 CRAWLER

Crawlers are also called robots, spiders, worms, wanderers, walkers, and know bots. The First crawler, Wanderer was developed by Matthew Gray in 1993 [31]. Due to the competitive nature of the search engine business, the designs of these crawlers have not been publicly described. Crawlers are basically programs that automatically traverse the web, retrieve pages and from the part of visited pages build a repository as shown in figure 2.4.

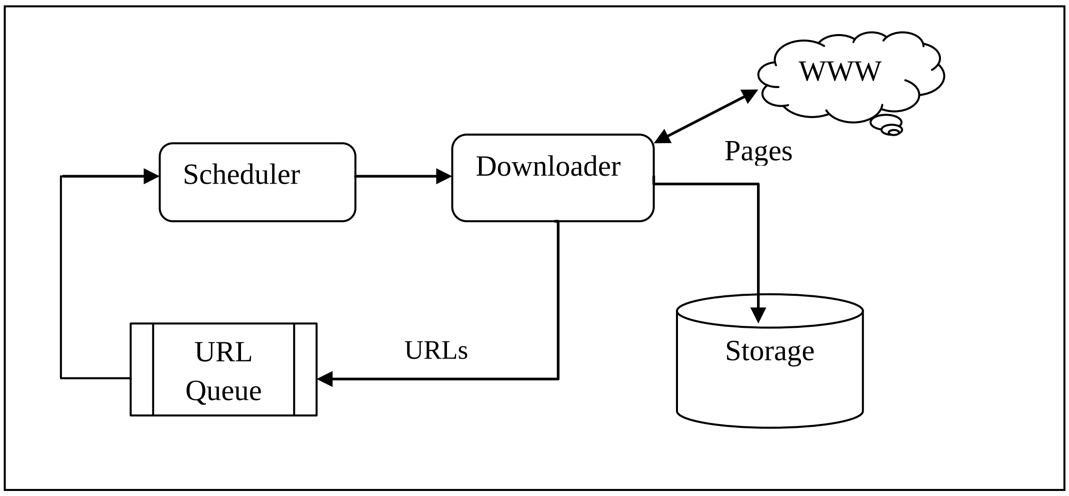


Figure 2.4 Web Crawler Architecture

A Crawler starts by taking an initial list of URL’s to visit called seeds. These seeds are initially added to an empty data structure which is either stack or queue depending on whether depth first search or breadth first search traversal is applied respectively.

From each URL, crawler finds other linked URL's in the page and after downloading those links store them also in repository.

Apart from adding new URL's to the data structure (URL queue) for downloading a crawler also performs some ordering to these links based on their importance. Web crawlers can copy all the pages they visit for later processing by a search engine. Search engine indexes those downloaded pages so that users can search them much more quickly.

The behaviour of a Web crawler is the outcome of a combination of policies:

- a selection policy that states which pages to download,
- a re-visit policy that states when to check for changes to the pages,
- a politeness policy that states how to avoid overloading Web sites, and
- a parallelization policy that states how to coordinate distributed web crawlers.

Crawlers continue visiting the Web, until local resources, such as storage, are exhausted. Once the search engine, has been through at least one complete crawling cycle, it may be informed by several indexes that were created during the earlier crawls. The algorithm for a typical crawler is shown in figure 2.5.

```
Crawler ()
    Step 1. read a URL from the set of seed URLs;
         2. determine the IP address for the host name;
         3. download the Robot.txt file that carries downloading permissions
            and also specifies the files to be excluded by the crawler;
         4. determine the protocol of underlying host like http, ftp, gopher
            etc.; based on the protocol of the host, download the document;
         5. identify the document format like doc, html, or pdf etc.;
         6. check whether the document has already been downloaded or not;
         7. if the document is fresh one
                read it and extract the links to the other cites from that document;
            else
                continue;
         8. convert the URL links into their absolute URL equivalents;
         9. add the URLs to set of seed URLs;
```

Figure 2.5 Algorithm of Basic Crawler

A crawler identifies a document from its URL, it picks up a seed URL and downloads corresponding Robot.txt file, which contains downloading permissions and the information about the files that should be excluded by the crawler. On the basis of the host protocol, it downloads the document and stores the related pages in its database. It then repeats the whole process. .

2.5 TYPES OF CRAWLER

The crawler has many types of traversing algorithm. Based on their working the crawler may be categorized in many types. Brief discussions of some are given in next sections.

2.5.1 Incremental Crawler

An incremental crawler [32, 41] updates an existing set of downloaded pages instead of restarting the crawl from scratch each time. This involves some way of determining whether a page has changed since the last time it was crawled. A crawler, which will continually crawl the entire web, based on some set of crawling cycles. The incremental crawler continuously crawls the web, revisiting pages periodically. During its continuous crawl, it may also purge some pages in the local collection, in order to make room for newly crawled pages.

There are basically two goals for an incremental crawler first is to keep the local collection fresh and second is to improve quality of the local collection. For keeping local collection fresh the crawler uses some revisit policies in order to download the updated information. In order to improve the quality of data the less important page from the repository are deleted and new pages are added.

Operational model of an incremental crawler: When the crawler decides to crawl a new page, it has to discard a page from the collection to make room for the new page. Now crawler has to decide which page to discard. This selection/discard decision is termed as the refinement decision. This refinement is dependent upon the importance of pages. To measure importance, the crawler can use a number of metrics, including Page Rank and Authority. The discarded page should have the lowest importance in the collection in order to ensure quality[61].

Now talking about freshness of the repository there are many algorithms developed by researchers the basic idea is decide on which page to update and when the updation is required. Here the idea is that instead of visiting a new page, the crawler may decide to visit an existing page to refresh its image. This is termed as update decision. To estimate how often a particular page changes, the Update Module records the checksum of the page from the last crawl and compares that checksum with the one from the current crawl. From this comparison, the Update Module can tell whether the page has changed or not.

2.5.2 Parallel Web Crawler

As the size of the Web grows, it becomes more difficult to retrieve the whole or a significant portion of the Web using a single process. Therefore, many search engines often run multiple processes in parallel to perform the above task, so that download rate is maximized. This type of crawler is known as a parallel crawler. Junghoo Cho [33] has suggested a general architecture of a parallel crawler which used multiple crawling processes. Each crawling process performed the same task as is done by a normal crawler i.e. they downloaded pages from the Web, stored them in repository, extracts URLs from them and followed those hyperlinks.

Then came Parallel Crawler using Augmented Hypertext Documents **(**PARCAHYD) [34] where parallelization of crawling system was done in order to download documents in a reasonable amount of time. It was a scalable parallel crawler. The following issues are significant in a parallel crawler:

- **Overlap**: When multiple processes run in parallel to download pages, it is possible that different processes download the same page multiple times. One process may not be aware that another process has already downloaded the page. Clearly, such multiple downloads should be minimized to save network bandwidth and increase the crawler’s effectiveness.
- **Quality:** Often, a crawler wants to download “important” pages first, in order to maximize the “quality” of the downloaded collection. However, in a parallel crawler, each process may not be aware of the whole image of the Web that they have collectively downloaded so far. For this reason, each

process may make a crawling decision solely based on its own image of the Web (that itself has downloaded) and thus make a poor crawling decision.

- **Communication bandwidth**: In order to prevent overlap, or to improve the quality of the downloaded pages, crawling processes need to periodically communicate to coordinate with each other. However, this communication may grow significantly as the number of crawling processes increases.

While challenging to implement, a parallel crawler has many important advantages, compared to a single-process crawler:

- **Scalability**: Due to enormous size of the Web, it is often imperative to run a parallel crawler. A single-process crawler simply cannot achieve the required download rate in certain cases.
- **Network-load dispersion**: Multiple crawling processes of a parallel crawler may run at geographically distant locations, each downloading “geographically-adjacent” pages. For example, a process in Germany may download all European pages, while another one in Japan crawls all Asian pages. In this way, we can disperse the network load to multiple regions. In particular, this dispersion might be necessary when a single network cannot handle the heavy load from a large-scale crawl.
- **Network-load reduction***:* In addition to the dispersing load, a parallel crawler may actually reduce the network load. For example, assume that a crawler in North America retrieves a page from Europe. To be downloaded by the crawler, the page first has to go through the network in Europe, then the Europe-to-North America inter-continental network and finally the network in North America. Instead, if a crawling process in Europe collects all European pages, and if another process in North America crawls all North American pages, the overall network load will be reduced, because pages go through only “local” networks.

2.5.3 Focused Crawler

A focused crawler is a type of topic sensitive crawler, which returns pages based on a given topic. It takes as input related web pages and tries to find out similar pages on the web, by r following hyper links [30, 35]. The topics are specified not using keywords, but using exemplary documents. The focused strategy is based on an assumption that a page under a certain topic (or region) is likely to be linked from another page with the same topic.

The focused crawler should return all similar pages while downloading minimum number of irrelevant documents. The goal of a focused crawler is to selectively find out pages that are related to a pre-defined set of topics. The focused crawler saves the network bandwidth and various resources by concentrating on making a crawl boundary so as to avoid irrelevant regions [36, 56].

A focused crawler has two main components:

(a) one is to determine if a particular web page is relevant to the given topic or not

(b) second is to find out how to proceed from a known set of pages.

Advantages of the focused crawling includes i) Higher Concentration of Relevant Pages by extracting only those pages which are specific to a particular topic the concentration of relevant pages is increased .ii) Efficiency in Crawling –The crawler saves both the network bandwidth and resource utilization by limiting the number of pages to be downloaded. iii) Reducing ambiguity by tracing down links only when there was a keyword found in the page, only the pages that are linked by another relevant page are collected. This will reduce ambiguity, since the page under consideration is guaranteed to be linked from the page related to the topic.

2.5.4 Distributed Crawler

In distributed crawling there are a number of crawler entities, which run on distributed sites and interact with each other equally. Distributed web crawling is a distributed computing technique where search engines engage many computers distributed among various places to index web pages. These systems allow users to offer their own computing and bandwidth resources for crawling web pages. By spreading the load across many computers, costs of maintaining large system is reduced.

The architecture of the distributed crawler [37] has been partitioned into two major components - *crawling system* and *crawling application*. The crawling system itself consists of several specialized components, in particular a crawl manager, one or more downloader's, and one or more DNS resolvers. All of these components, plus the crawling application, can run on different machines (and operating systems) and can be replicated to increase the system performance.

The crawl manager is responsible for receiving the URL input stream from the applications and forwarding it to the available downloader's and DNS resolvers while enforcing rules about robot exclusion and crawl speed. A downloader is a high-performance asynchronous HTTP client capable of downloading hundreds of web pages in parallel, while a DNS resolver is an optimized stub DNS resolver that forwards queries to local DNS servers.

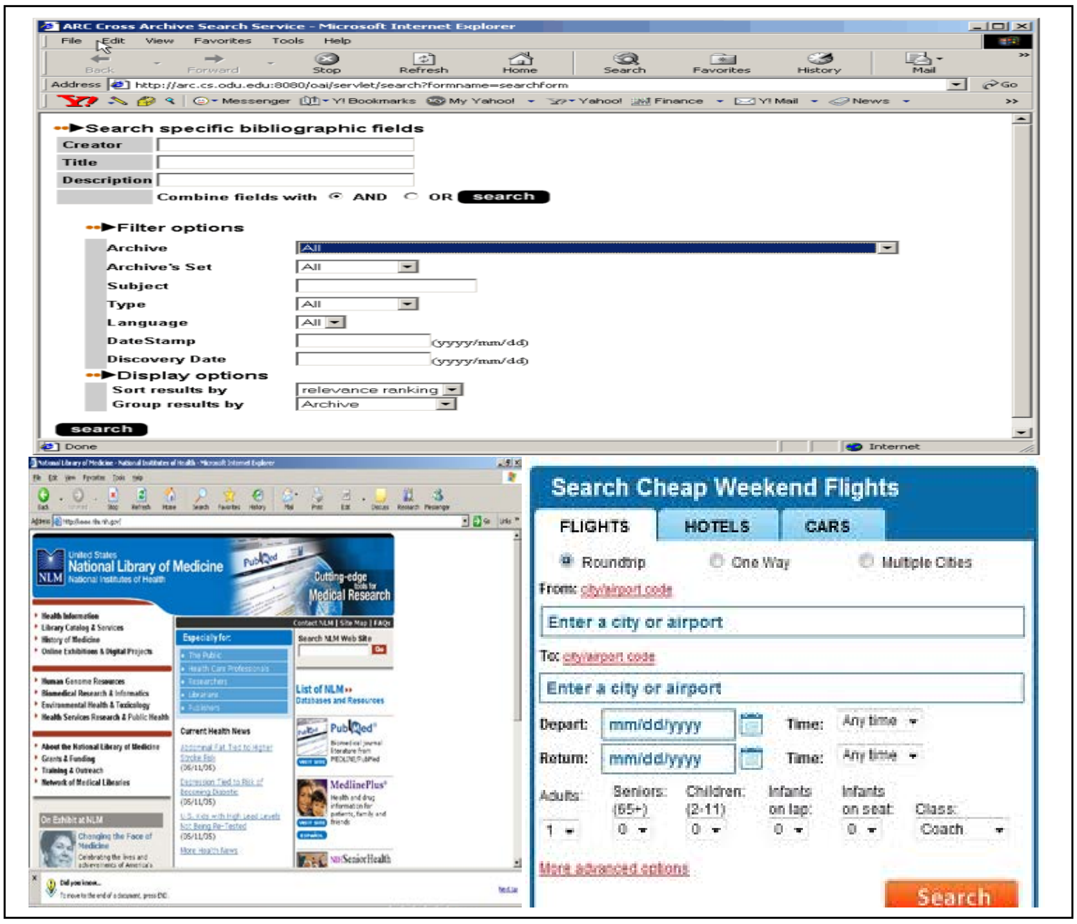
2.6 HIDDEN WEB

Hidden web is that part of World Wide Web which cannot be directly accessed by simple search mechanism or by surfing from one page to another following a hyperlink. It is hidden because user has to fill in the form interfaces to extract the particular information. The part of the web which can be directly accessed by simply following hyperlinks on the web pages is termed as Surface Web. A large part of the Web is hidden behind search forms and is reachable only when users type in a set of keywords, or queries, to the forms. These pages are often referred to as the Hidden Web or the Deep Web [8, 9, 10].

Examples of Hidden Web from interfaces are shown in figure 2.6. They can be:

1. Certain file formats (PDF, Flash, Office files, and streaming media) because they aren't HTML text.
2. Most real-time data (stock quotes, weather, airline flight info) because this type of data is transient & storage intensive.
3. Dynamically generated pages (cgi, JavaScript, asp, or most pages with "?" in URL) because the simple crawler cannot create queries to fire for generating dynamic web pages.
4. Web accessible databases because crawlers can't type and fill forms for retrieving these databases.

Public information on the hidden web is currently around 500 times larger than the surface web as stated by Bergman [8]. It has also been stated that there are around 96000 Hidden Web sites which contain 7500 terabytes of data.



Address http://arc.cs.odu.edu:8080/oai/servlet/search?formname=searchform

Search specific bibliographic fields

Creator	
Title	
Description	
	Combine fields with [✓ AND] OR search

Filter options

Archive	All
Archive's Set	All
Subject	
Type	All
Language	All
DateStamp	(yyyy/mm/dd)
Discovery Date	(yyyy/mm/dd)

Display options

Sort results by	relevance ranking
Group results by	Archive

search

[✓ Roundtrip] One Way Multiple Cities

From: city/airport code — Enter a city or airport

To: city/airport code — Enter a city or airport

Depart: mm/dd/yyyy Time: Any time

Return: mm/dd/yyyy Time: Any time

Adults:	Seniors: (65+)	Children: (2-11)	Infants on lap:	Infants on seat:	Class:
1	0	0	0	0	Coach

Figure 2.6 Examples of Hidden web interfaces.

As the volume of hidden information grows, there has been increased interest in techniques that allow users and applications to leverage this information. It is not easy to access this high quality information because this hidden web data is not indexed by normal crawler which follows hyperlink structure. In hidden web a large amount of data is generated dynamically by servers with the help of backend databases [48]. These dynamically generated pages are user dependent and information on these pages changes very quickly.

A survey done by Madhvan in [38] stated that there are around 647000 hidden web resources identified by Google's index. In UIUC [39] survey 3,00,000 hidden web databases and 4,50,000 query interfaces have been found.

Figure 2.7 shows the distribution of various types of contents in overall hidden web. It shows that Topical databases i.e. large internal site documents and archived publications make up nearly 54% of all Hidden Web, whereas about 13% of Hidden Web sites are related to internal sites (including shopping sites with auctions and classifieds) and 11% are publication sites. The remaining categories collectively comprise remaining 22% of hidden Web sites[9, 14].

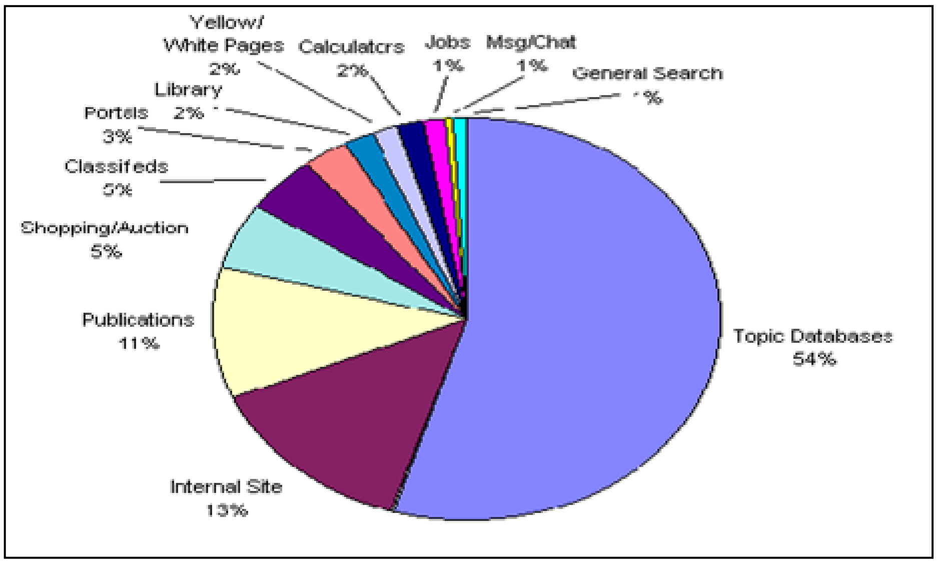


Figure 2.7 Hidden Web Content Distribution

The major information resources stored in the invisible web are normally in non-textual formats and free content-rich databases created by Government agencies, educational institutions and other organizations around the world. They include patents, digital exhibits, laws, dictionaries, phone books, people finders, items in web stores or auctions, multimedia and geographical files. Further, the information is usually new and dynamically changing in content such as news, job postings, flight schedules, accommodation reservation, stock prices, etc.

Simple crawlers are not capable of discovering and indexing these pages because of the following reasons:

1. Simple crawlers follow basic hyperlink structure to download pages but there are no static links to hidden web pages.
2. Hidden web has structured as well as the large volume of unstructured data that need to be indexed.

3. The only entry point to hidden web site is a query interface; multi attribute query interfaces have more than one attribute and require their respective values to be submitted by user.
4. Also web pages are created dynamically by firing user query; they cannot be indexed by traditional search engines. The large amount of high quality information is buried under dynamically generated sites.

For these reasons the Research is in progress on how to "get into” the hidden web and extract the information.

2.7 HIDDEN WEB CRAWLER

Hidden Web Pages are accessible only after submitting queries. There is no direct link available for such kind of web so, typical crawlers are not able to find them. Hidden Web Crawler is one that can crawl and extract content from the hidden web databases. Such crawlers are enabled with indexing, analysis and mining of hidden web contents. These crawlers have to interact with the query form in order to find the contents of Hidden Web. Apart from this these crawlers are used to classify and categorise hidden database based on the extracted contents. In order to access hidden web these special crawlers go through a query interface and after filling fields value that interface form is submitted which in turn generate a response page. This response page makes the crawler able to access the linked urls which enables the search engines to perform indexing and make this possible for further access.

The only way to extract hidden web information is by inputting the values for each field in the form interface which is termed as the entry point to Hidden Web pages; there are two main issues that need to be resolved to implement an effective Hidden Web crawler. These are

i) The crawler has to be able to understand and model a query interface, and

ii) The crawler has to come up with meaningful queries to issue to the query interface.

2.7.1 Basic Architecture of a Hidden Web Crawler

A basic Hidden Web Crawler consists of four components as shown in figure 2.8 and are described as follows:

i) Internal Form Representation which is internal representation of search form and it is built from form page.

ii) Task Specific Database is encompassed with every Hidden Web Crawler and this database contains all the necessary information required to formulate search queries relevant with the particular task.

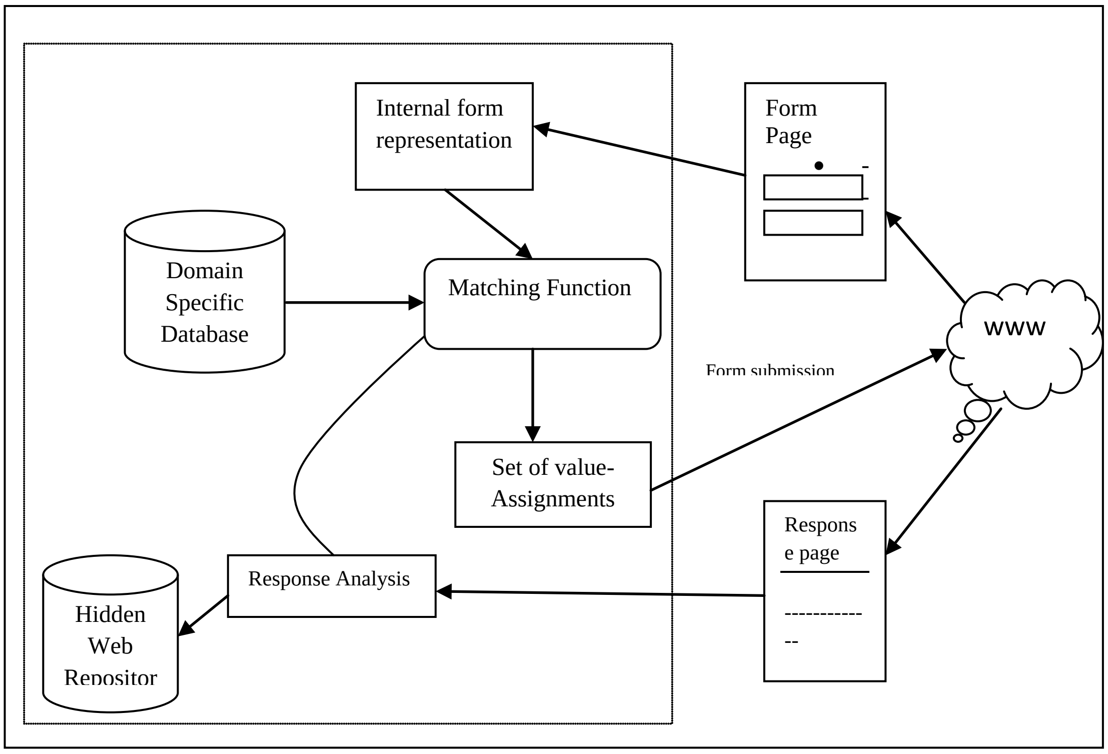


Figure 2.8 Architecture of Hidden Web Crawler

iii) Matching Function takes as input, an internal form representation & the database contents and as an output it produces a set of value assignment. Basically this function associates each element in the internal form representation to a value, which is used to fill-up and submit the form.

iv) Response Analysis stores the generated response page into crawler's repository. In addition to this it distinguishes between pages containing error messages and search results. It also analyses the feedback which helps in tuning the matching function and updating the set of value assignment.

Hidden web crawler [19] performs the following sequence of actions for each identified form on a page:

Step 1: Parse and process the form to build an internal representation.

Step 2: Generate the best values to be assigned to the various form elements and submit the completed form using the assignment.

Step 3: Wait for response pages.

Step 4: Analyze the response page. Report errors if any, else use the received information as feedback for next iteration. If the response page contains hypertext links, follow them immediately and recursively, to a pre-specified depth.

It may be noted that the links in the response page are also added to the URL queue. However, for ease of implementation, the response pages are also navigated immediately and that too, only up to a depth of 1.

2.8 TYPES OF HIDDEN WEB CRAWLER

In order to download the Hidden Web contents from the WWW the crawler needs a mechanism for Search Interface interaction i.e. it should be able to download the search interfaces, automatically fill them and submit them to get the Hidden Web pages. Many researchers are trying to develop novel ideas to access hidden web in order to improve searching experience for users [43, 54, 55].

A brief overview at few of them is given in the following section:

i) Hidden Web Exposer (HiWE): Raghavan and Garcia-Molina proposed HiWE [19], a task-specific hidden-Web crawler, the main focus of this work was on learning Hidden-Web query interfaces. A prototype hidden Web crawler called *HiWE (Hidden Web Exposer)* was developed at Stanford. The architecture of HiWE is shown in figure 2.9.

HiWE automatically processes, analyses, and submit forms by using an internal model for forms and their submissions. A new technique named as layout-based information extraction (LITE) is used to extract useful information. The first advantage of HiWE is that it was task specific and the crawler was extracting only relevant pages. The automatic form filling was there but with some human assistance.

Talking about limitations first is that the HiWE"s is not able to recognize and respond to simple dependencies between form elements (e.g., given two form element corresponding to states and cities, the values assigned to the „city‟ element must be cities that are located in the state assigned to the „state‟ element).

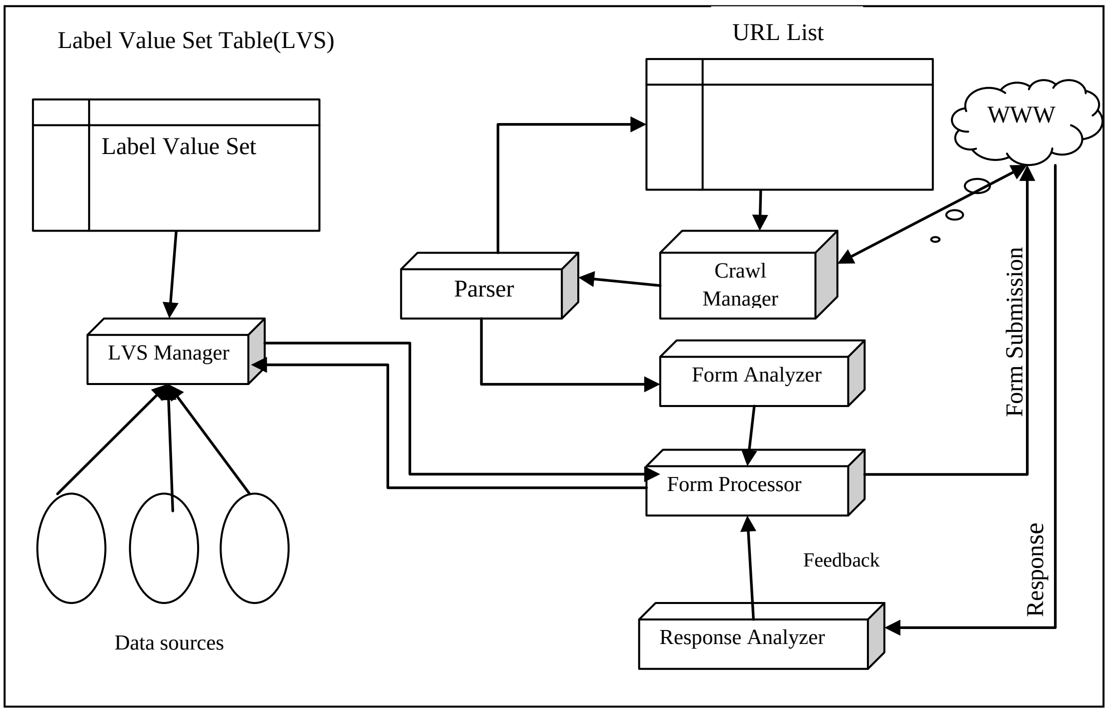


Figure 2.9 Architecture of HiWE Crawler

The second limitation is HiWE‟s lack of support for partially filling out forms; i.e., providing values only for some of the elements in a form.

ii) Query Generation for Downloading Hidden Web Content: Ntoulas et al. [12] differ from the previous studies, that, it provided a theoretical framework for analyzing the process of generating queries. In his work it was concluded that the only "entry" to Hidden Web pages is through querying a search form, there are two core challenges to implementing an effective Hidden Web crawler: (a) The crawler has to be able to understand and model a query interface, and (b) The crawler has to come up with meaningful queries to issue to the query interface.

The first challenge was addressed by Raghavan and Garcia-Molina in, where a method for learning search interfaces was presented. Here, they gave solution to the second challenge, i.e. how a crawler can automatically generate queries so that it can discover hidden information.

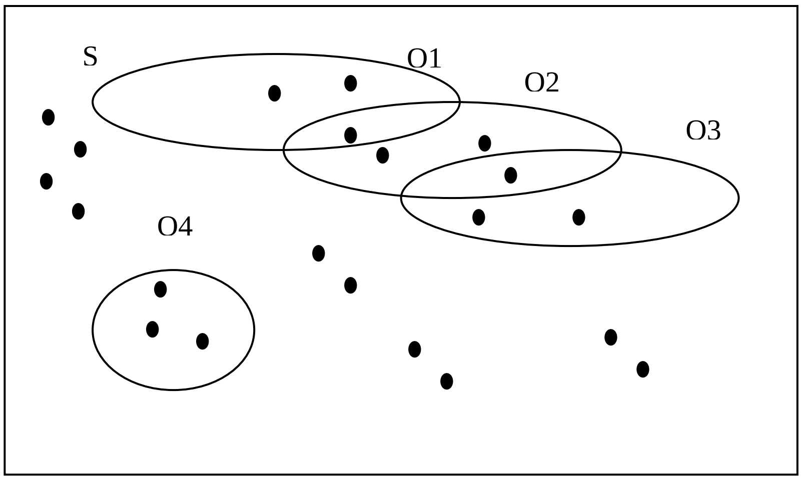


Figure 2.10 Set Formalization for Query Selection.

By predicting which query to fire next depending on the current query was the main motive of the work. They performed set formalization technique for query selection shown in figure 2.10. This body of work was often referred to as database selection problem over the Hidden Web. The disadvantage was that the work only supported single attribute queries.

iii) Adaptive Crawler for Locating Hidden Web*:* Barbosa and Freire [17] experimentally evaluated methods for building multi-keyword queries that could return a large fraction of a document collection. New adaptive crawling strategies to efficiently locate the entry points to hidden-Web sources were proposed. A framework was designed as shown in figure 2.11 whereby crawlers automatically learn patterns of promising links and adapt their focus as the crawl progresses.

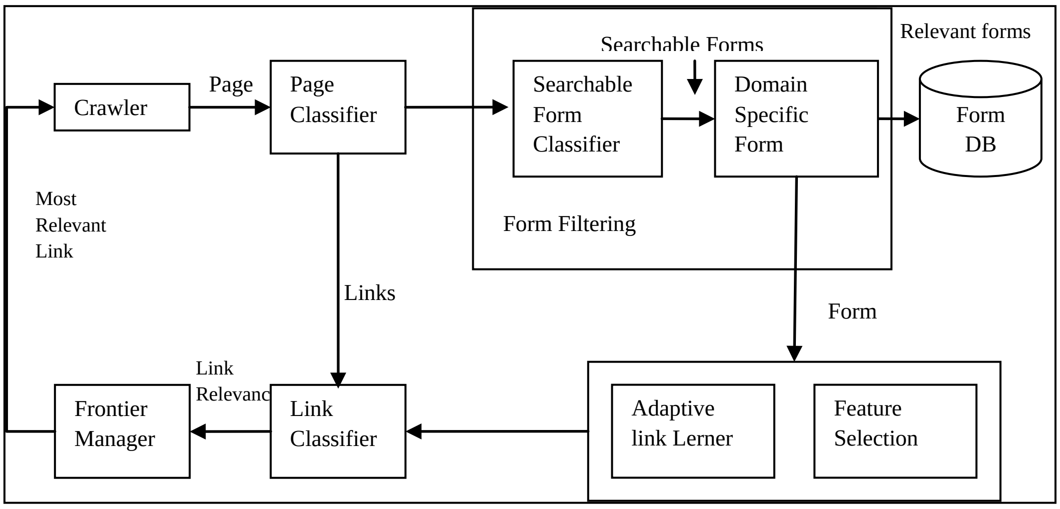


Figure 2.11 Architecture of Adaptive Crawler for Locating Hidden Web.

This strategy effectively balances the exploration of links with previously unknown patterns, making it robust and able to correct biases introduced in the learning process.The disadvantage is that the link classifier here used takes the training data manually hence is not exhaustive and very time consuming.

iv) Hidden web data Extraction using Wrappers: Lage et al.[20] claimed to automatically generate agents to collect hidden web pages by filling HTML forms. They introduced the concept of web wrappers in this research. To extract the unstructured data from web, wrappers are used. Wrapper is nothing but a set of programs which takes a set of target pages from the web source as an input. These set of target pages are extracted by "Spiders" which automatically crawls the web for web pages. Hidden web agents assist the wrappers to deal with the data available on the hidden web. The advantage of this technique is that it can access a large number of web sites of diverse domains. However, the limitation of this technique is that it can access only that web site that follows common navigation pattern.

v) Google's DeepWeb Crawl: Madhvan et al.. in 2009 discusses the approach used by Google in filling Web forms [38]. HTML forms usually offer more than one input and hence a layman's strategy of enumerating the Cartesian product to identify of all possible inputs can result in a very large search space. They have presented an algorithm (figure 2.12) that appropriately chooses the input combinations so as to efficiently navigate the search space by including only those generated URLs which seem suitable for inclusion in the web search index.

```
GetInformativeQueryTemplates (W: WebForm)
            I: Set of Input = GetCandidateInputs(W)
candidates: Set of Template = { T: Template | T.binding = {I}, I2 I}
informative: Set of Template = while (candidates 6= )
newcands: Set of Template = foreach (T: Template in candidates)
              if ( CheckInformative(T, W) )
                   informative = informative [ { T }
                   newcands = newcands [ Augment(T, I)
                   candidates = newcands
return informative
Augment (T: Template, I: Set of Input)
     return { T' | T'.binding = T.binding [ {I},I2 P, I 62 T.binding }
```

Figure 2.12 Algorithm of Incremental Search for Informative Query Templates (ISIT)

The first step of the approach contributes the in formativeness test for evaluating the query templates, i.e., combinations of form inputs. The basic idea of the in formativeness test is that all templates are probed to check which can return sufficiently distinct documents. The next step develops an algorithm that efficiently traverses the space of query templates to identify the ones suitable for surfacing. A template that returns enough distinct documents is deemed a good candidate for crawling. As a last step the approach contributes to an algorithm which predicts appropriate input values for the various form fields. They have described how the identification of typed inputs in web forms (e.g. zip codes, dates, prices) contributes to better results. The main advantage is that it efficiently navigates the search space of possible input combinations by selecting query templates. Limitation is that the efficiency of crawling technique was not considered.

vi) A Framework for Domain-Specific Interface Mapper (DSIM)*:* Komal Kumar Bhatia [13]: Here a domain specific crawler for the hidden web, DSHWC that considers multi-input search forms has been developed. The working of DSHWC has been divided into several phases with the first one concerning the automatic downloading of the search forms. Phase 2 describes the most important component Domain-specific Interface Mapper that automatically identifies the semantic relationships between attributes of different search interfaces and guides the next step of merging the interfaces so as to form a Unified Search Interface (USI). The USI produced thereof is filled automatically and submitted to the Web as shown in figure 2.13.

After obtaining response pages, the DSHWC stores the downloaded pages into Page repository that maintains the documents crawled/updated by the DSHWC along with their URLs. DSHWC is a fully automated crawler which aims to obtain the response pages from Hidden Web by submitting filled form pages.

Advantages of DSHWC: 1) Multi-strategy interface matching 2) use of mapping knowledge base to avoid repetition for minimizing the mapping effort 3) Enhances the scope of developing a specialized search engine for the Hidden Web.

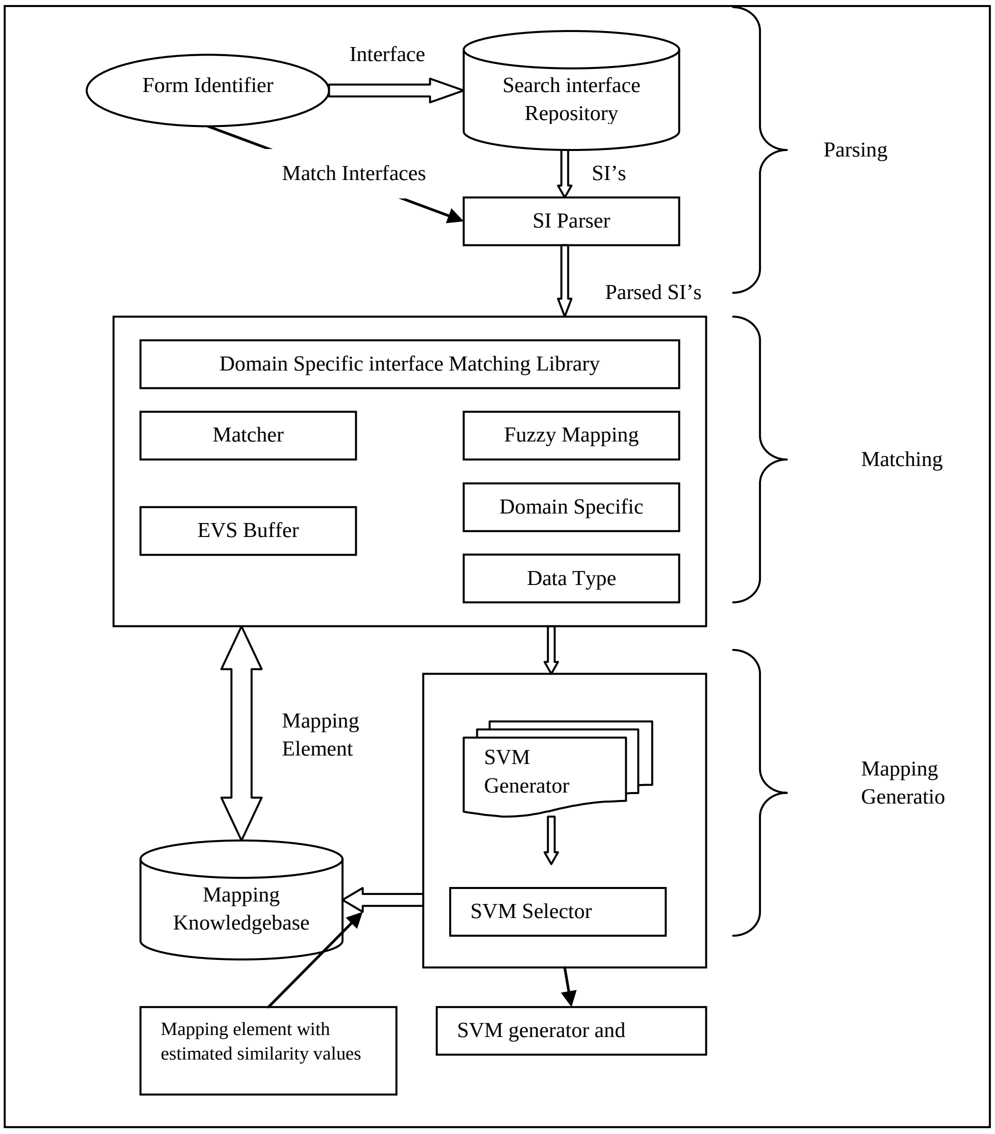


Figure 2.13 Architecture of DSHWC

Limitations of DSHWC: 1) Indexing technique was not specified for storing pages in the repository. 2) Defined the performance only for crawling while the efficiency of schema matching and merging procedures over variety of query interfaces has not been quantified.

vii) A Framework for Incremental Hidden Web Crawler: Rosy et al. [41] a framework has been proposed that updates the repository of search engine by re-crawling the web pages that are updated more frequently. It uses a mechanism for adjusting the time period between two successive revisits of the crawler based on probability of the web page.

A comparison of different techniques based on some important attributes is given in Table 2.1.

Table 2.1 Comparison of Different Hidden Web Crawlers

Techniques	**Support for Structured docs**	**Unstructured**	**Simple Search Interface**	**Classification**	**Query Probing**	**Use of Ontology**	**Dynamic revisit**
Raghavan et al.	Yes	No	Yes	Yes	No	No	No
Ntoulas et al.	Yes	No	Yes	No	Yes	No	No
Barbosa and Freire	Yes	Yes	Yes	Yes	No	No	No
Madvan et al.	Yes	Yes	Yes	No	Yes	No	No
Lage et al.	Yes	Yes	Yes	No	Yes	No	Yes
Komal Kumar Bhatia	Yes	Yes	Yes	No	No	No	No
Rosy et al.	Yes	Yes	Yes	No	Yes	No	Yes

It can be easily observed from the above comparison table that some of the crawlers are supporting unstructured documents. Also none of the hidden web crawlers developed so far use ontology to find appropriate values for form interfaces.

2.9 EXPLORING THE NEED OF ONTOLOGY IN RETRIEVING HIDDEN WEB

Traditional crawlers do not help in retrieving this hidden web data and search results given by traditional crawlers have low precision. This scenario leads to requirement of new crawler that can perform search process efficient and avoid visiting large irrelevant portions of web. New crawler termed as hidden web crawler should be able to download the search interfaces, automatically fill them and submit them to get the hidden web pages. By combining the hidden web retrieval problem with domain specific ontology, we automatically fill in the text boxes with values of instances present in ontology. Table 2.2 gives a comparison between basic search system and the system developed using ontology.

Table 2.2 Comparison Table between Basic Search and Ontology Based Search

	Basic Search	Ontology Based Search
Relevancy	Low	High
Effort of User Navigation	High	Low
Semantics	Not Used	Semantics used
Understanding of User Requirement	Low	Better than Simple search
Space Requirement	Low	High
Cost of Implementation	Low	High
Time to respond	Low	High

Retrieving hidden web using ontology mainly consists of following major tasks in proposed work:

1. Generating ontology for a particular domain.
2. Creating database of ontology.
3. Developing a hidden web crawler.
4. Mapping of interface with ontology for filling up the interfaces.
5. Indexing and Ranking of the data retrieved

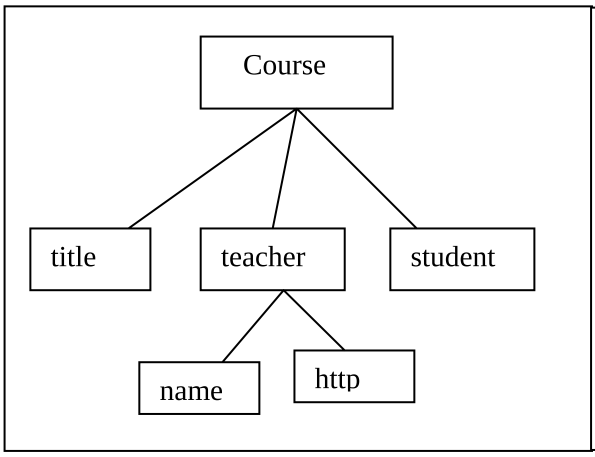
2.10 ONTOLOGY

Ontology[46] defines a common vocabulary that is used as information source for any specific domain. It defines a domain with objects or concepts and their properties and relations with each other. Ontology is used for knowledge sharing and reuse, and also for representing knowledge in building “intelligent” applications.

Before describing more about ontology we must know the background why it has been developed and how it is better than previous concepts of HTML and XML in specifying web pages.

2.10.1 XML

XML is already widely known in the Internet community, and is the basis for a rapidly growing number of software development activities. It is designed for markup in documents of arbitrary structure, as opposed to HTML, which was designed for hypertext documents with fixed structures. XML uses tags embedded in the content of a document to delimit and label parts of the document, and those parts are known as XML elements. The start and end tags include an XML element type name and may also contain XML attributes.



```
<course>
   <title>...</title>
        <teacher>...</teacher>
              <name>...</name>
              <http>...</http>
            <students>...</students>
</course>
```

Figure 2.14 Example Showing XML Tags and Hierarchal Structure of the same

An XML attribute is a pair made up of an attribute name and an attribute values. Multiple XML attributes can occur within the start tag of an element, but each start tag can contain only one XML attribute with a given attribute name. XML attribute values can contain only character data.

A well-formed XML document creates a balanced tree of nested sets of open and close tags, each of which can include several attribute-value pairs as shown in figure 2.14. There is no fixed tag vocabulary or set of allowable combinations, so these can be defined for each application.

In XML 1.0 this is done using a document type definition to enforce constraints on which tags to use and how they should be nested within a document. A DTD defines a grammar to specify allowable combinations and nestings of tag names, attribute names, and so on.

XML Document Type Definitions (DTDs) and XML Schemas provide means of describing/defining constraints on the structure of a class of XML documents, the structural relationships that can exist between components i.e. XML Schemas and XML DTDs describe content models for named XML element types and attributes. An XML document which conforms to the rules of the XML specification and to the structural constraints described by an XML DTD or XML Schema is described as valid.

Although XML schema offer several advantages over DTDs, their role is essentially the same: to define a grammar for XML documents. XML is used to serve a range of purposes:

1. Serialization syntax for other markup languages. The DTD is useful because it facilitates a common understanding of the meaning of the DTD elements and the structure of the DTD.
2. Semantic markup of Web pages. An XML serialization (such as the example above) can be used in a Web page with an XSL style sheet to render the different elements appropriately.

3. Uniform data-exchange format. An XML serialization can also be transferred as a data object between two applications.

It is important to note that a DTD only specifies syntactic conventions; any intended semantics are outside the realm of the XML specification.

2.10.2 RDF

The Resource Description Framework (RDF)[50] set of specifications describe a means of constructing simple statements about resources. The Resource Description Framework is a recent W3C recommendation designed to standardize the definition and use of metadata—descriptions of Web-based resources. However, RDF is equally well suited to representing data.

Central to RDF is the idea of the resource, which can be anything you wish to describe - a document, a physical object, a person, an imaginary being, a concept, anything at all - and the idea of identifying resources using Uniform Resource Identifiers (URIs). The basic building block of the RDF data model is the triple, consisting of a subject, a predicate and an object as depicted in figure 2.15 below. The subject is a URI reference (or a "blank node"), the predicate is a URI reference, and the object is a URI reference, a blank node or a literal.

The following triples represent three statements, each one stating a relationship between two resources:

Subject	Predicate	Object
http://example.org/doc/123	http://purl.org/dc/elements/1.1/creator	http://example.org/person/manvi
http://example.org/person/manvi	http://xmlns.com/foaf/0.1/name	"Manvi Siwach"
http://example.org/person/manvi	http://xmlns.com/foaf/0.1/knows	http://example.org/person/pramesh

Figure 2.15 Example showing <S,P,O> triple

2.10.3 RDFS

- RDF gives a language for meta data annotation, and a way to write it down in XML, but it does not provide any way to structure the annotations
- RDF Schema augments RDF to allow you to define vocabulary terms and the relations between those terms
  - it gives “extra meaning” to particular RDF predicates and resources
  - e.g., Class, subClassOf, Property, domain, range
- These terms are the RDF Schema building blocks (constructors) used to create vocabularies

Just as XML schema provides a vocabulary-definition facility, RDF schema lets developers define a particular vocabulary for RDF data and specify the kinds of object to which these attributes can be applied. In other words, the RDF schema mechanism provides a basic type system for RDF models. This type system uses some predefined terms, such as Class, subPropertyOf, and subClassOf, for application-specific schema. RDF schema expressions are also valid RDF expressions (just as XML schema expressions are valid XML). The triples in above example of figure 2.15 can be represented in RDF/XML as shown in figure 2.16.

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:dc="http://purl.org/dc/elements/1.1/"
         xmlns:foaf="http://xmlns.com/foaf/0.1/">
  <rdf:Description rdf:about="http://example.org/doc/123">
    <dc:creator rdf:resource="http://example.org/person/manvi"/>
  </rdf:Description>
  <rdf:Description rdf:about="http://example.org/person/manvi">
    <foaf:name>Manvi Siwach</foaf:name>
    <foaf:knows rdf:resource="http://example.org/person/pramesh"/>
  </rdf:Description>
</rdf:RDF>
```

Figure 2.16 RDF of Example Showing <S,P,O> Triple of Figure 2.15

- RDFS defines the ontology
  - classes and their properties and relationships
  - what concepts do we want to reason about and how are they related
  - there are authors, and authors write books
- RDF defines the instances of these classes and their properties
  - Mark Twain is an author
  - Mark Twain wrote “Adventures of Tom Sawyer”
  - “Adventures of Tom Sawyer” is a book

Notation: RDF(S) = RDF + RDFSIn order to exchange RDF data between applications, the data must be represented in some digital format. This process is referred to as serialisation. The RDF data model is independent of any specific serialisation syntax. In particular RDF does not rely on XML.

Table 2.3 A Comparison Table of RDF/RDFS/OWL Languages

Characteristics	RDF (Resource Description Framework)	RDFS (RDF Schema)	OWL (Ontology web language)
Level of expressivity	Lowest level to build ontologies and cannot express more complex relations.	RDFS extends RDF with schema vocabulary and can better express more relation properties as compared to RDF but less rich than OWL.	It is topmost level to build ontologies or schema on the top of RDF dataset.
Degree of Inferencing	RDF cannot define special meaning to vocabulary such as type, subclassof relationships.	RDF Schema allow to define vocabulary terms and the relations between those terms, it gives “extra meaning” to particular RDF predicates.	OWL adds more semantics to the schema. It allows to specify far more about the properties and classes.
Richness in Tag Property	RDF provides few predicates to define classes relations i.e rdf:type, rdf:class.	RDFS provides a new predicate rdfs:subClassOf, rdfs:subPropertyO, rdfs:domain and rdfs:range	OWL provides owl:disjointwith, owl:FunctionalProperty, owl:InverseFunctionalPropert, owl:inverseOf, owl:Sym-metricProperty
Validity of Ontology	RDF defines way how to write stuff, however it may sometimes be wrong and not a valid ontology.		OWL defines way to write stuff and its specification defines what exactly can be written with RDF in order to have valid ontology.

It can be concluded that OWL[51] provides better way to represent knowledge and relationships between various entities of knowledge. OWL provides various functions to express and infer.

2.10.4 Defining Ontology

Definition: Ontology is an explicit representation of concepts of some domain of interest,with their characteristics and their relationships[15].

More formally speaking, “Ontology is a formal explicit description of concepts in a domain of discourse (classes called concepts), properties of each concept describing various features and attributes of the concept (slots sometimes called roles or properties), and restrictions on slots (facets called role restriction). Ontology together with a set of individual instances of classes constitutes a knowledge base.

Sharing common understanding of the structure of information among people or software agents is one of the common goal in developing ontology by Musen 1992, Gruber 1993[44, 46].

An example of book domain ontology is shown in figure 2.17. All the oval shaped nodes are classes and subclasses whereas the square shaped nodes are relationship between the classes.

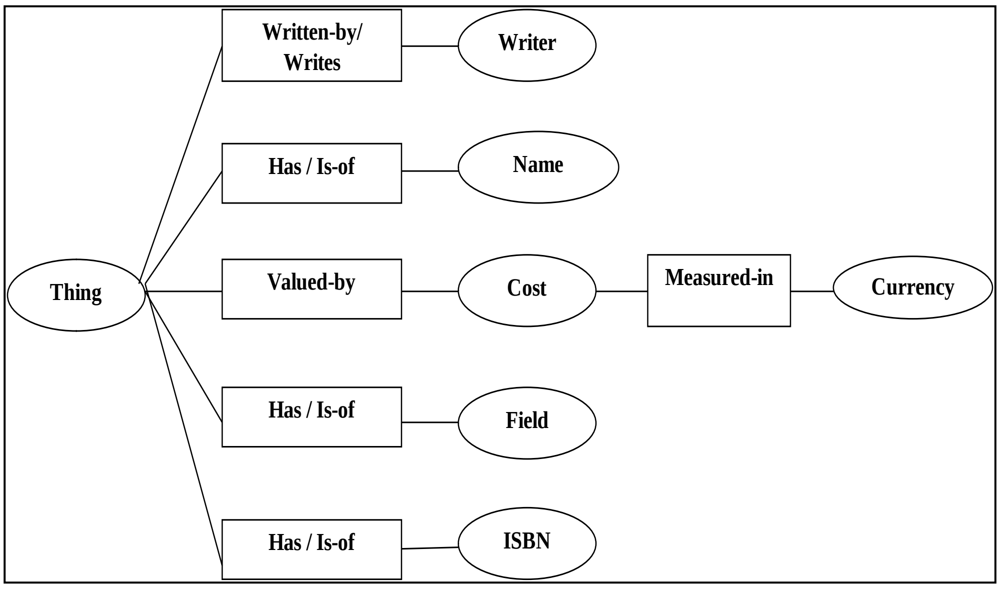


Figure 2.17 An Example of Developed Book Ontology Developed

Ontology is one of the best ways for creating Domain knowledge that has common understanding of the structure of information among people. Ontology technology allows arbitrary user-defined relationships among classes and allows adding properties to relationships such as symmetry, transitivity, and inversion. These properties are used in reasoning that’s why Ontology supports inference process in knowledge base. Ontologies can play a crucial role in enabling Web-based knowledge processing, sharing, and reuse between applications. Generally defined as *shared formal conceptualizations of particular domains*, ontologies provide a common understanding of topics that can be communicated between people and application systems.

Following are the few usages of developing an Ontology:

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

Ontologies can be very useful in improving the process of information retrieval [57, 62] in two ways:

1. It allows to abstract the information and represent it explicitly- highlighting the concepts and relations and not the words used to describe them.

2. Ontologies can possess *inference functions*, allowing more intelligent retrieval. For example a "basketball player" is also a "professional athlete", and an ontology that defines the relations between these concepts can retrieve one when the other is queried.

3. For using Ontology in information retrieval systems and applications several research activities have been done in the area of ontology construction because it provides a richer knowledge representation that improves machine interpretation of data.

i) Ontology Components: Common components of ontology include:

(i) Individuals: It may refer as instances or objects.

(ii) Classes: sets, collections or concepts.

(iii) Attributes: aspects, properties, features, characteristics.

(iv) Function terms: complex structures formed from certain relations that can be used in place of an individual term in a statement.

(iv) Restrictions: formally stated descriptions of what must be true in order for some assertion to be accepted as input.

(v) Rules: statements in the form of an if-then (antecedent-consequent) sentence that describe the logical inferences that can be drawn from an assertion in a particular form.

(vi) Events: The changing of attributes values or relation.

ii) Domain Ontology

Domain-specific ontology models a specific domain, which represents part of the world. Particular meanings of terms applied to that domain are provided by domain ontology.

The general stages in the design and development of ontology are shown in figure 2.18 and are summarised as follows [47]:

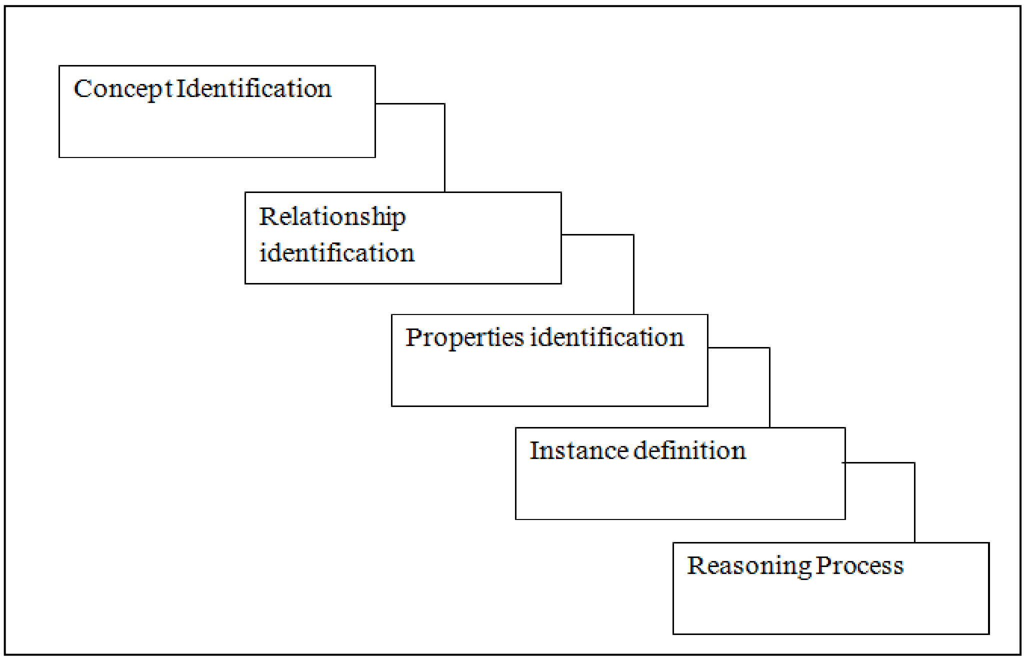


Figure 2.18 Steps for creating Ontology

1. Concept Identification

The knowledge acquisition activity is done with the requirements specification phase. The knowledge acquisition was extremely important, because this activity defines the extent to which we can create ontology. The knowledge acquisition, the first step of the building process, begins with the reading and selection of domain source. The knowledge sources had different views about subjects, but they all agreed on the separation of concepts. The first step to organize the concepts was this structured knowledge.

2. Relationship and Properties Identification

During conceptualization the acquired knowledge was structured. The domain is sub divided into sub domains, then module for each sub domain is created and proper interfacing is to be done. The main activities of the conceptualization in the development of each module are

1. Classification of groups of concepts in classification trees.
2. Description of properties.

3. Identification and definition of instances:

After specifying all the classes and subclasses the instances which are the object of classes are defined. One can create a knowledgebase with the help of ontology. For creating knowledge base the instance values are stored and queries are fired.

4. Reasoning

After creating Ontology of domain we may find relationship, attribute values and can infer any reasoning about that domain. The different tools have different query format where user can ask a question and get the desired result.

Protégé [49] IDE is capable of generating, for example, OWL or RDF. In addition to this capability, the creation of a database with Domain information transforms the formalized model into an implemented one. There are examples of it in the current version of the ontology. One problem that came up is a consequence of dividing the ontology into modules.

2.10.5 Ontology Development Tools

Ontology tools can be used for generating, validating and maintenance of domain specific knowledge, So that we can further use this knowledge base for information sharing and information retrieval. The language that can be used for creating ontology for any domain are XML, RDF, DAML-OIL and OWL and these are used to maintain the metadata of ontology.

Various tool for ontology development are available but first we have to analyse and evaluate them before use. Tools are protégé, oiled, Ontolingua, Onto Edit, ICOM, RDF edt and Web ODE [58,75].

Table 2.4 shown below give details about comparative study and analysis of various tools.

Table 2.4 Comparison of Various Ontology Development Tools

	Import Format	**Export Format**	**Graph view of ontology**	**Reasoning and consistency Check**	**User Environment**	**Support for web services**	**Ontology merging**
Protégé	XML, RDF(S), XML Schema	XML, RDF(S), XML Schema, Java html	Via plug-ins like GraphViz	Via plug-ins like PAL and FaCT RacePro , hermit	Limited (Multi-user capability added to it in latest version)	Via Protégé-OWL plug-in	Via Anchor-PROMPT plug-in
Oiled	RDF(S), OIL, DAML-OIL	RDF(S), OIL, DAML OIL	No	Via FaCT	No	Very limited namespace	No
Onto lingua	IDL, KIF	KIF,CLIPS, IDL, Prolog syntax	No	Via Chimaera	Via write only locking,	Yes	No
Onto Edit	XML, RDF(S), FLogic and DAML+OIL	XML, RDF(S), FLogic and DAML + OIL	Yes	Yes	No	Yes	No
RDF Edit	RDF(S), OIL, DAML, SHOE	RDF(S), OIL, DAML, SHOE	No	Only checks writing mistakes	No	Via RSS RDF Site Summary	No
Web ODE	RDF(S), UML, DAML+	RDF(S), UML, DAML	Form based graphical	Yes	By synchronization ,	Yes	Via ODE merge

We used various criteria like Import Format, Export Format, Graphical view of ontology, User environment, support for web services, Consistency check and Reasoning and ontology merging.

1. Import Format: This defines the languages supported by tool by loading any external ontology.
2. Export Format: This defines the languages supported by tool while exporting any ontology file.
3. Graphical View of Ontology: Tools support various plug ins to support ontological structure in graph format.
4. User Environment: It means number of users that can support tool at a time make incremental changes in ontology and notify others about the modification.
5. Support for web services: Tools that provide direct access to knowledge base stored in web.
6. Consistency Check and Reasoning: Feature of tool that maintain consistency in knowledge base and allow Reasoner tool for inference from knowledge base.
7. Ontology Merging: It is a method of bringing two conceptually divergent ontology.

It can be concluded from the comparison table that Protégé provides graph view of ontology by graphviz plugin. Also it can be seen that protégé supports web services and can be easily imported and exported. Using protégé reasoning can be done with help of SPARQL [52] query language. Most importantly it is open source and can be easily installed on single machine.

2.11 ONTOLOGY MAPPING

Given two ontology's O1 and O2, mapping one ontology onto another means that for each entity (concept, relation or instance) in Ontology O1, a corresponding entity, which has the same intended meaning, in Ontology O2 is found.

It is not possible in real world to provide a same name for a concept or object having same meaning also it may be possible that two different names point to same concept, so there is always a gap between expressing knowledge by two individuals. Humans can very well map this difference.

But in order to make machine understand this gap, we need to bridge the knowledge gaps between different systems during their communication. Ontology mapping serves to bridge this knowledge gap. When ontologies are mapped, applications can query data from different data sources transparently; applications can treat each data source the same irrespective of their differing underlying representations.

An example of ontology mapping is shown in figure 2.19 below. Here two ontology are mentioned both belongs to same (book) domain. As it is clearly depicted from the diagram some concepts in ontology1 like “Author”, “Title” and “Price” are equivalent to “Writer”, “Name” and “Cost” respectively from the ontology2.

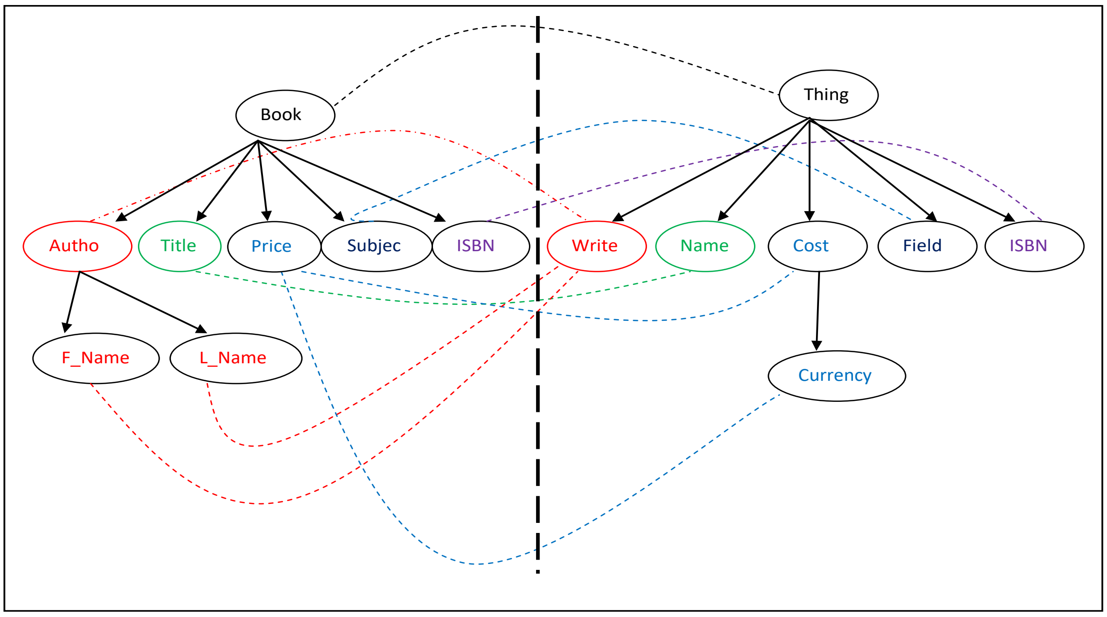


Figure 2.19 Sample Ontology Mapping

Majority of ontology mapping tools developed so far are used to finds a one-to one corresponding mapping between concepts in two ontology. These mapping tools are classified into two categories: source-based and instance-based.

In source-based mapping tools, the similarity of the concepts (based on the concept properties) and the structure of the ontology (as described in source ontology) are compared. Examples of source-based mapping tools are PROMPT [70], Chimaera [McGuinness et al., 2000], and ONION [73]. PROMPT and Chimaera merge two source ontologies into a new one. The merged ontology contains concepts from both sources. They compare similarity of concept names to generate a match list of concepts. Users decide which concepts should be mapped based on the match list.

In instance-based mapping tools the similarity of the concepts based on the source ontologies and their data instances are compared. Examples of instance-based ontology mapping tools are FCA-Merge [Stumme & Madche, 2001] and GLUE [71]. FCA-Merge merges two source ontologies into a new ontology. FCA-Merge generates a pruned concept lattice by analyzing the frequencies of usage of concepts. Merging decisions are made based on the pruned concept lattice. FCA-Merge suits best the mapping of text documents: it requires a set of common instances for the mapping ontologies. For example, the instances are in the form of documents or homepages. GLUE gives a set of pairs of related concepts with some certainty factor associated with each pair. It analyzes the distributions of the concepts in data instances of the source ontologies and uses joint probability distribution to calculate the similarity between two concepts. GLUE, however, does not consider the structure of the ontologies (i.e., the relationships between concepts) during mapping.

Once the task of ontology mapping is done corresponding terms (node) in one ontology whose values are known are used to make entries into the fields of query interface. After successful submission of query interface form a response page is generated and this makes possible to access the hidden web.

2.11.1 Ontology Mapping Techniques

Different approaches to the matching problem have been proposed in the literature, for example [64, 65, 66, 71, 73] of the past approaches.

1) **Doan et al. [71] developed GLUE** with the recognition that many measures of similarity can be employed to determine mappings between ontologies. They articulate three goals for their approach: first, the notions of similarity employed in the system should be well defined so as to enable evaluation and the application of, "special-purpose techniques"; second, similarity measures should correspond to some intuitive notion of similarity relying primarily on the semantic content of the concepts presented rather than their syntactic attributes; and finally, the authors recognize that many useful similarity measures exist, each being

appropriate to particular situations, and that the best approach is to be able to employ the most appropriate measures where needed.

2) **Prasenjit Mitra et. al. developed ONION** [73] it was one of the first tools for ontology merging. In their tool ONION the authors use rules and inferencing to execute mappings, but is based on manually assigned mappings or very simple heuristics. ONION results in a set of mappings (articulation rules using their terms) between two ontologies. It transforms source ontologies into graphs. The nodes and the edges are used to match two graphs. Nodes are matched based on their names and a set of user-defined synonyms words.

3) **Marc and Steffen proposed Quick Ontology mapping (QOM)** [65] which is very efficient in terms of time taken to perform mapping but it suffers from poor mapping quality. Since this approach largely focuses on instance matching which cannot be provided, they refrained from running the tests on a poorly trained estimator which would immediately result in poor quality results. The architecture for the same is shown in figure 2.20.

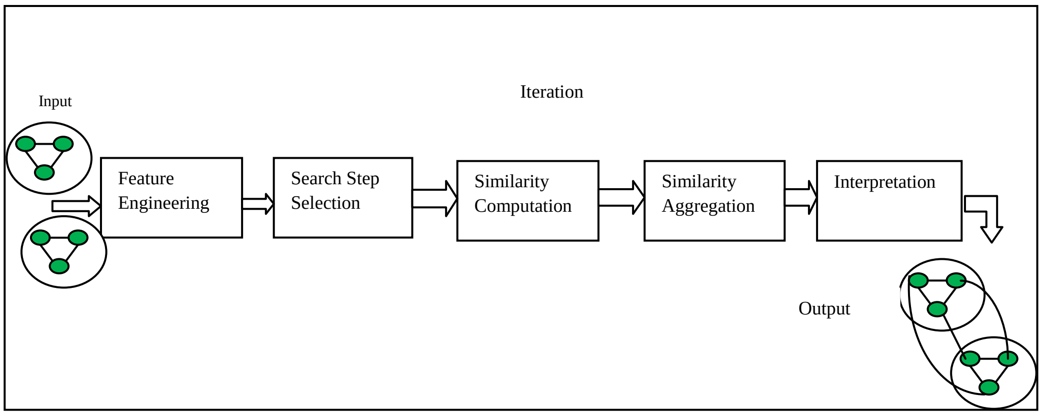


Figure 2.20 Architecture of QOM Ontology Mapping Technique

In addition to this, QOM works well only with those ontologies that are having a specialized terminology and its matching accuracy decreases when mapping ontology with more general terminologies.

4) **A Concept hierarchy based Ontology Matching Approach by Ying, Liu and Bell** [66]**:** They proposed such a model that may contain complex mapping results, i.e. several entities in ontology(O_1) map to the same entity in ontology(O_2).In this approach even after the completion of mapping process

additional work has to be done for selecting the best mapping pair among multiple matching pair for a single.

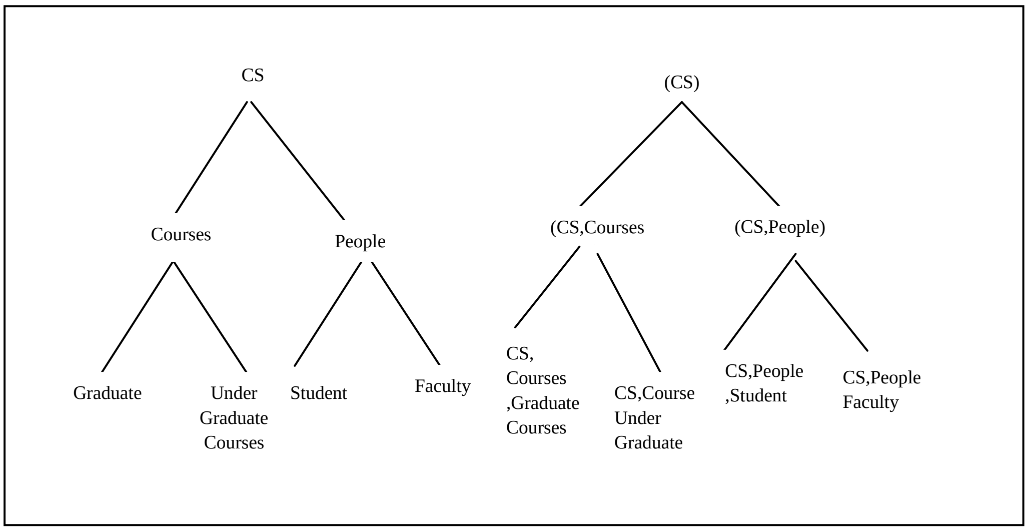


Figure 2.21 Concept hierarchy based Ontology Matching Approach

This post-matching process makes this approach inefficient in terms of time taken to perform the entire mapping process node.

5) **PROMPT: Algorithm and Tool for Automated Ontology Merging and Alignment by Noy and Musen, 2000**[70]: It compares all entity pairs based on their labels. In a postprocessing step an acknowledgement of the user becomes necessary. –The consequence are results with a lower quality. PROMPT, Chimaera and ONION use similarity between concept names for mapping. They work well for ontologies having a specialized terminology like medical ontology where each concept is a disease and each disease has a unique name. Their matching accuracy decreases when mapping ontologies with more general terminologies.
6) **A mapping framework for distributed ontologies (MAFRA)** [72]**:** They developed a multi-strategy process that calculates similarities between ontology entities using different algorithms. The first strategy focuses on acquiring a lexical similarity between each entity in source entity with each and all entities in target entity. Subsequently, a next step calculates the so called property similarity, that is responsible to acquire the similarity between concepts based on their properties, either attributes or relations. The bottom-up similarity intends to propagate the

similarity (or dissimilarity) from lower parts of the taxonomy to the upper concepts. It uses the property similarity as input and propagates the values to the top. This similarity gives a good overall view of similarity between taxonomies.

Table 2.5 A Comparison of Ontology Mapping Techniques

	GLUE	**MAFRA**	**QOM**	**ONION**	**PROMPT**
Input	Two taxonomies with their data instances in ontologies	Two ontologies	Two list of terms from two ontologies	Terms in two ontologies	Two input ontologies
Output	A set of pairs of similar concepts	Mapping of two ontologies by the semantic bridge ontology	A list of matched pairs of terms with score ranking similarity	Set of Articulation rules between two ontologies	A merged ontology
User Interaction	User-defined mappings for training data , similarity measure, setting up the learner weight, and analyzing system's match suggestion	The domain expert interface with the similarity and semantic bridging modules and it has graphical user interface	It requires human validation at the end of the process.	A human expert chooses or deletes or modifies suggested matches using a GUI tools	The user accepts, rejects or adjusts System suggestions.
Mapping strategy	Multi-strategy Learning approach (machine learning technique)	Semantic bridge	Lexical similarity whole term, word constituent, synset, and type matching.	Linguistic Structure matcher, inference based heuristics	Heuristic based analyzer.
Domain knowledge	Yes	Yes	No	Yes	Yes

2.12 INDEXING THE HIDDEN WEB

Indexing is a way used to represent information from a document so that those conducting searches can retrieve the information easily. Without an index, the search engine would scan every document in the corpus, which would require considerable time and computing power. Ontology based web indexing system attaches different class (concept) to the keyword and does a mapping between keyword and ontology class. But having attached the concept does not solve the whole problem. Using WordNet context with every keyword in which it is used is attached. Due to attachment of context with keyword, more relevant results are retrieved in optimized way in lesser time response to a search query.

Ding et al. [76] introduces a double indexing mechanism for search engines, which means, it has document index as well as word index. The so-called document index is based on the documents clustering, and ordered by the position in each document. During the retrieval process, the search engine first gets the document id of the word in the word index, and then goes to the position of corresponding word in the document index. The mechanism proposed by them seems to be time consuming as the index exists at two levels.

Mahapatra et al. [77] proposes an inverted indexing system. Here indexing words are mapped to their location in document. The index is sorted by its keys and works well with Boolean operators (AND, OR, NOT). **Silvestri et al. [78]** proposes the reordering algorithm which partitions the set of documents into k ordered clusters on the basis of similarity measure. According to this algorithm, the biggest document is selected as centroid of the first cluster. The process keeps on repeating until all the k clusters are formed. The drawback is that the biggest document may not have similarity with any of the documents but still it is taken as the representative of the cluster. **Zamir et al.[79]** proposes the threshold based clustering algorithm in which the number of clusters is unknown. However, two documents are classified to the same cluster if the similarity between them is below a specified threshold. This threshold is defined by the user before the algorithm starts.

Table 2.6 A Comparison Table of Various Indexing Techniques

Indexing Technique	Methodology	Advantages	Limitations
Double indexing mechanism, Ding et.al. [76]	The index is based on the documents clustering, and ordered by the position in each document.	Simple and relevant indexing mechanism.	Time consuming as the index exists at two levels.

Inverted indexing, Mahapatra1 et.al. [77]	This technique has terms marked as keys and these terms are mapped to the document they appear in. The index is sorted by its keys and works well with Boolean operators.	1. Very simple 2.No complexity involved	It can only tell whether a word occurs in a document or not. It can't tell how often it occurs or its location.
Document identifiers to enhance Compressibility, Silvestri et. al. [78],	This technique gave a reordering algorithm to compress the inverted index.	1 Due to compression index takes less space. 2. The algorithm has linear complexity.	1.No new algorithm given to improve index 2. No involvement of meaning or semantics of keywords.
Fast and Intuitive Clustering of Web Documents, Zamir et. al. [79]	A clustering methods that intersect the documents in a cluster to determine the set of words (or phrases).	Fast and quick navigation through the results.	Theortical framework given only, no experimentation done. Preprocessing for clustering is required
Context based Indexing using Ontology, Parul et.al. [80]	1. Thesaurus is used to obtain the context in which term is used. 2.Index has three columns a)Context (eg apple is fruit and apple is i-phone) b)Term c)Doc id	1. As Context is stored with every term so results relevant to users are retrieved.	1.As index is linear time required in retrieving result is O(n). 2. One word may belong to different concepts this work didn't focus on that.
Ontology based Indexing using BST tree, Pooja et.al. [81]	Same as above except Index is not stored in linear array. It is stored in the form of Binary Search tree.	Better than Linear array as Best case time complexity: O(logn)	As all the terms(node) are not at same level time to retrieve result in Worst case: (n)
Ontology based Indexing using AVL Tree, Nidhi et.al. [82]	Same as above Except Index is stored in form of AVL tree	Better than BST Time Complexity is O(n) in all cases. As all leaf nodes are at same level.	Every node can have only maximum two children.

From the above table it can be concluded that indexing scheme starting from simple keyword to ontology. Some of the techniques tried to attach classes using ontology but none of the techniques gave evaluation supporting the same. Also the ambiguity in various terms were not resolved.

2.13 RANKING TECHNIQUES IN HIDDEN WEB

To present the results to the user in an ordered manner, Page Ranking methods are applied, which can arrange the results in order of their relevance, importance and content score[83,84]. Search engines use two different kinds of ranking factors: Query-dependent factors and Query Independent Factors to calculate the rank of a web page. Query-dependent factors are all ranking factors that are specific to a given query, while query-independent factors are attached to the results, regardless of a given query. Query-dependent factors used measures word documents frequency, the position of the query terms within the result page or the inverted document frequency, that are used commonly in any basic search engine. Some of the query independent factors are Link popularity, Click popularity and up to-datedness of the page etc. The aim of this section is to comparatively analyze the existing ranking algorithms or techniques for hidden web pages.

A) Content Based Hidden Web Ranking Algorithm (CHWRA) [85]: In this paper proposed a ranking algorithm which consists of four different attributes. These are: a) Page Rank , b) Term Weighting Technique [TWT], c) User's Feedback, d) Visitor Count.

a) Page Rank: The PageRank component checks the entire link structure of the network and calculates the PR value of web page and redistributes it to the links within the web page. The formula used in calculation of pagerank value is given as:

$$PR(A) = (1-d) + d\,[\,PR(T1)/C(T1)+...+\quad PR(Tn)/C(Tn)\,]$$

Where PR(A) – pagerank value of page A
d – is the damping factor, set to 0.85
T1….Tn – set of pages pointing to A
PR(Tn) – pagerank of page Tn
C (Tn) – number of hyperlinks pointing out from page Tn.

b) **Term Weighting Technique:** The term weighting technique is based on probabilistic and vector space model. There are three main parameters used in calculating TWT. The parameters are document length, document frequency and term frequency.
c) **User's Feedback:** This method takes user's feedback into account in the form like and dislikes count. Like and dislike count are taken as the positive or negative response respectively to the web page and affects the popularity of the web page, thus affecting the rank value of the web page.

d) **Visitor Count:** In this method hits on the web page are considered as the visitor count. It is assumed that more the number of hits on the web page and higher the popularity of the web page.

Limitations of CHWRA

- The page rank algorithm is commonly used by the conventional search engines. It is not effective for Hidden Web pages.
- Some fraud websites knowingly add a lot of popular keywords which are not related to the content in the title or the content of the page to cheat search engines.
- This technique do not emphasis on ranking of hidden web pages rather it does ranking of general surface web pages.
- This technique uses user feedback and visitor count for ranking but it didn't explained well how these are taken into consideration.

B) SCUM: A Hidden Web Page Ranking Technique [87] proposed ranking of Hidden Web pages using three steps as follows:

a) Structure Page Rank Calculation
b) Content Page Rank Calculation
c) Usage Page Rank Calculation

- **Structure Page Rank Calculation***:* The web pages from the WWW are highly connected. Graph databases are used, the nodes represent the entities (web pages) and edges represents the relationships (here inlinks and outlinks). The graph is created by using NEO4J and cypher query language. The pattern recognition is done very easily in the case of graph database. For ex: it is very easy to extract the pattern that the node no. 699 of domain car receives all the inlinks from the web pages of different domain i.e. property. Interconnection among web pages is shown in fig. below. Most of the newbie websites gave their advertisements on the high rank popular websites. Mostly the domains of these websites are far apart from each other. The popular websites get financial benefit from it. So they promote the newbie websites. These new websites then are able to share the page rank of the popular websites. So in order to avoid such type of page rank sharing the new technique should be developed which will avoid this incorrect page rank sharing policy. So new formulae for calculating the page rank. For calculating the rank of page “A” which assume that it receive “n” inlinks from same domain and “m” inlinks from different domains. The pagerank will be shared using the formulae given below.

Rank(s) A= (Rank(s) P1 / (OS P 1) + Rank (structure) P n / (OS P n)) if OSP1>0

Here Rank(s) A is structure rank of page A, OSP1 is the no of outlinks of web page P1 to web pages of same domain.

- **Content Page Rank Calculation**: The content mining is extraction of knowledge from text in the web pages. In this the content of extracted web pages will be analyzed and on the basis of the contents the pages will be ranked. The relevance of the page will be analyzed on the basis of the domain, the quality of content, spam detection. In order to make the content present on the web page machine understandable Resource Description file (RDF) is created for every web page . Query will be fired on the RDF and relevance and quality of data on the web page is calculated.

Rank(c)A = Relevance + Quality

Advantages of the SCUM

- It not only considers the links but also the contents of the web pages for the rank calculation.
- The interest of the users is also considered. So every user is able to see the pages of its own interest at the top.
- The algorithm makes use of all aspects of web mining to calculate the page rank.
- The proposed algorithm will rank the pages from the hidden web and surface both.

Limitations of SCUM

- This algorithm in structure rank calculation focus on inlinks and outlinks on the page. No. of inlinks are not relevant for calculating hidden web page ranking.

- In usage rank calculation emphasis is on particular user. Thus this algorithm is more user based but user priority may change with time.
- This algorithm is not efficient for hidden web page ranking.
- Proper design and implementation of algorithm is not given.

C) Deep-Web Search Engine Ranking Algorithm : Brian Wong in et al[88] gave ranking algorithm which utilizes best-fit scoring functions using quality factors and a dynamic weighting algorithm that changes the factor weighting based on user behavior. Search engine utilizes two factor scoring function to rank results – a combination of distance score d and referral score r. The distance score is inversely proportional to the physical distance between search result and location of interest. The referral score represent the popularity of the result amongst the searched websites.

Advantages

- This algorithm is scalable and requires minimal pre-processing to generate the factor weightings.
- It rank results considering user behaviour and requirements .
- This technique efficiently ranks the hidden web pages.

D) Rank Discovery over Hidden Web Databases [89] introduced problem of This paper define a comprehensive spectrum of ranking functions according to various dimensions such as query-dependent vs. static, observable vs. proprietary, and whether the scoring attribute can be queried or not. This paper discuss the feasibility of rank discovery for each type of ranking function, and show that different types of ranking functions require fundamentally different approaches for rank discovery. For proprietary and observable ranking functions, they developed RANK-EST(algorithm) which interleaves two separate procedures for handling high and low ranked tuples, respectively.

E) Trust and Profit Sensitive Ranking for the Deep Web and On-line Advertisements [90] considered the emerging problem of ranking the deep web data considering trustworthiness and relevance. In this paper end-to-end deep web ranking by focusing on: (i) ranking and selection of the deep web databases (ii) topic sensitive ranking of the sources (iii) ranking the result tuples from the selected databases has been discussed.

Table 2.7 A Comparison Table of Various Ranking Techniques

Ranking Algo	Use of Query Dependent factors	Use of Query Independent Factor	Technique Used	Relevancy to Hidden Web	Remarks
Content based hidden web ranking	NO	YES	Pagerank and Term Weighting Technique	LESS	This technique does not emphasise on ranking of hidden web pages.
SCUM: A Hidden Web Page Ranking Technique	NO	YES	Web structure, content and usage mining	MORE	
Deep-Web Search Engine Ranking Algorithm	YES	YES	Best-fit scoring functions using ten quality factors and a dynamic weighting algorithm.	MORE	This algorithm is scalable and requires minimal pre-processing to generate the factor weighting
Rank Discovery by Saravanan et al	YES	NO	Developed RANK-EST(algorithm) which interleaves two separate procedures for handling high and low ranked tuples,	PARTIAL	It discuss the feasibility of rank discovery for each type of ranking function.
Raju Balakrishnan et. al.	YES	NO	Various score functions are used.	MORE	

The table shows that few of the ranking techniques use query dependent factors, others use query independent factors. Only one technique is using both factors for calculating ranking of web. It can also be observed that the ranking techniques developed so far are not developed keeping hidden web in mind.

2.14 PROBLEM IDENTIFIED IN EXISTING APPROACHES

A critical look at the available approaches indicates the following issues need to be deal with towards building an effective ontology based information retrieval system for hidden web:

i) In present approaches the high quality data hidden behind form interfaces is extracted using basic label value matching techniques. These techniques are inefficient and lack in data relevancy. The hidden web data need to be extracted using new approach which involves attaching meaning to the data hence it support to get more relevant results.

ii) A very petite work has been done in direction of creating semantic database and using the same for extracting hidden web data.

iii) There is a need of automatic generation of ontology which is not addressed by any of the work.

iv) Ontology Mapping System which will map the form interface with predefined ontology are: a) Pairing irrelevant node for mapping. b) Lack of an effective mapping approach while matching between ontology c) Poor resource (mainly time) utilization.

v) The available systems are not fully automated, there is no work done towards synchronizing the processes of form filling, submitting, downloading the hidden web data.

vi)The indexing schemes developed so far do not take context of the term in to account reducing the efficiency in searching. They also cannot disambiguate the ambiguous terms.

vii) The ranking techniques so far have not been developed considering hidden web data. Specialized technique need to be developed for ranking hidden web data.

A design of a novel ontology based information retrieval system for retrieving hidden web deals with all the identified shortcomings/problems found in existing work. The architecture of the proposed system with brief discussion on each component is given in next chapters.

Chapter 3

ONTOLOGY BASED INFORMATION RETRIEVAL SYSTEM FOR HIDDEN WEB

3.1 INTRODUCTION

In order to download the Hidden Web contents from the WWW the crawler needs a mechanism for Search Interface interaction i.e. it should be able to download the search interfaces in order to automatically fill them and submit them to get the Hidden Web pages. Crawler must be so intelligent that it chooses the exact values to be filled. Secondly a common search interface should be provided to the user, where user type in the query and get the desired hidden web information efficiently without filling up the form interfaces manually.

Traditional hidden web crawler try to download this data based upon simple Label-Value matching mechanism which is not suffice for this high quality tremendous information lying behind the form interfaces available.

In order to synchronize the processes like form filling, submitting, downloading the web pages and indexing etc. for hidden web crawling, domain knowledge is required. This domain knowledge can be represented in the form of ontology. Ontology provides a common vocabulary of an area and defines, with different level of formality, the meaning of terms and relationships between them. By combining the hidden web retrieval with domain specific ontology, the proposed work automatically fills in the text boxes with values from the developed ontology. This will make not only the retrieval process task specific, but will increase the likelihood of being able to extract just the relevant subset of data.

3.2 DESIGN OF ONTOLOGY BASED INFORMATION RETRIEVAL SYSTEM FOR HIDDEN WEB

The objective of the work is to automate the process of searching, viewing, filling in and submitting the search forms followed by analysis of the response pages, along with a common search interface to the user. The architecture of the whole system is divided into four modules designed to separately handle the various groups of actions as listed.

i) Ontology Builder
ii) Hidden Web Crawler
iii) Ontology based Indexer
iv) Rank Calculator

These modules further contain various sub modules described as shown in figure 3.1

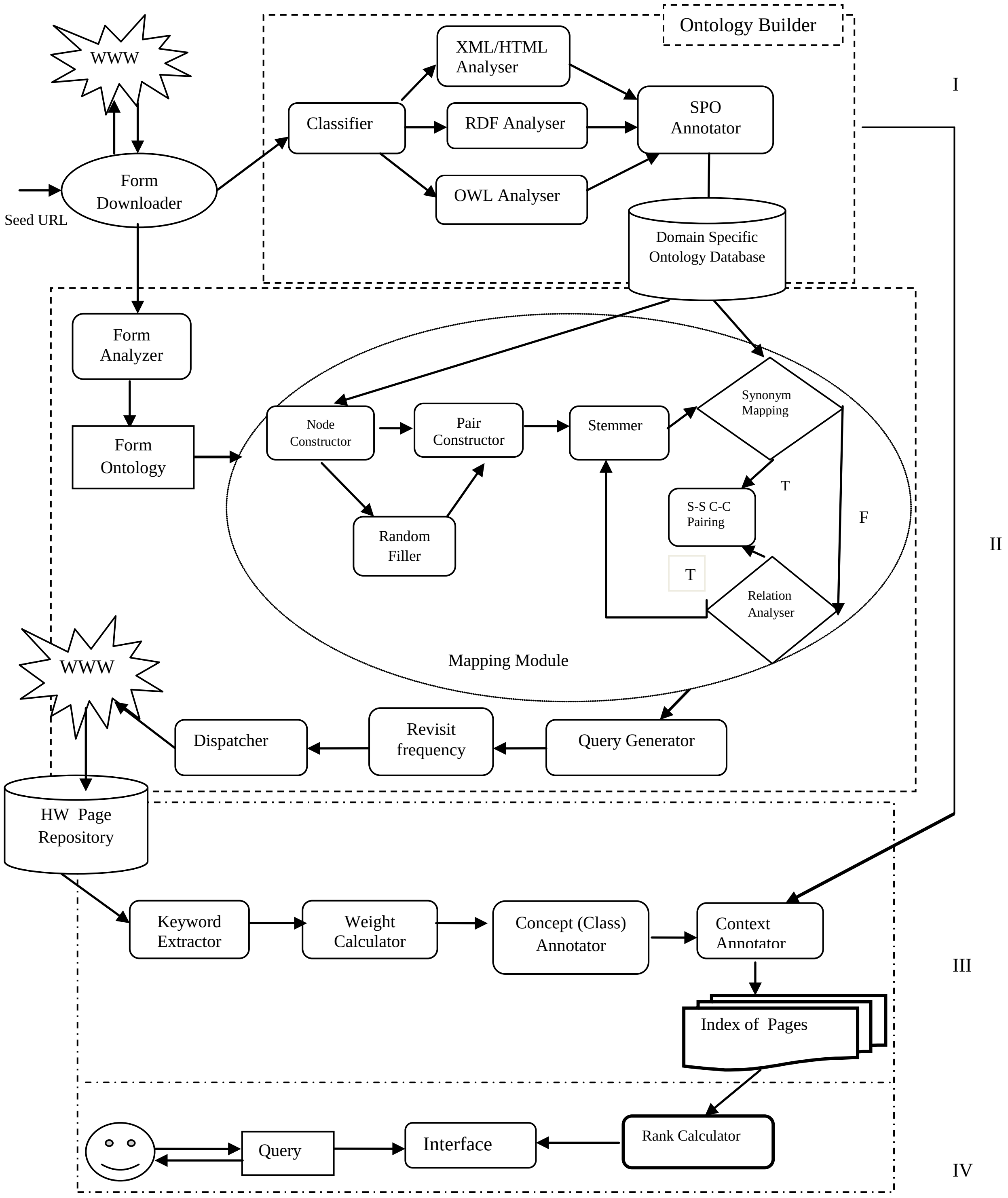


Figure 3.1The Proposed Architecture of Ontology based Information Retrieval System for Hidden Web

A brief discussion on each of the major modules is given below:

3.2.1. Ontology Builder: The first component of the system is generation of ontology which is described in this phase in detail. Firstly ontology has been designed and generated manually using Protégé for Book Domain by taking instance values from faculty of YMCA and their publications [94]. The basics of Class, Properties and Attributes has been described and explained in this module. Subject, Predicate, Object <S,P,O> triples has been defined for book domain as shown by figure 3.2. This ontology serves as a prototype for generating automatic ontology.

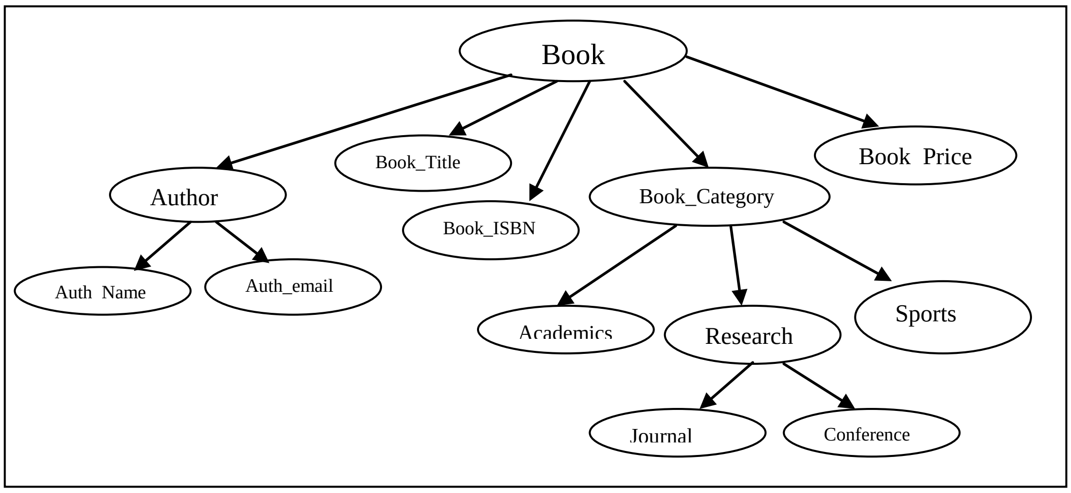


Figure 3.2 Class Hierarchy in Book Domain Ontology developed Using Protégé

The ontograph view of the developed book domain ontology is shown in figure 3.3.

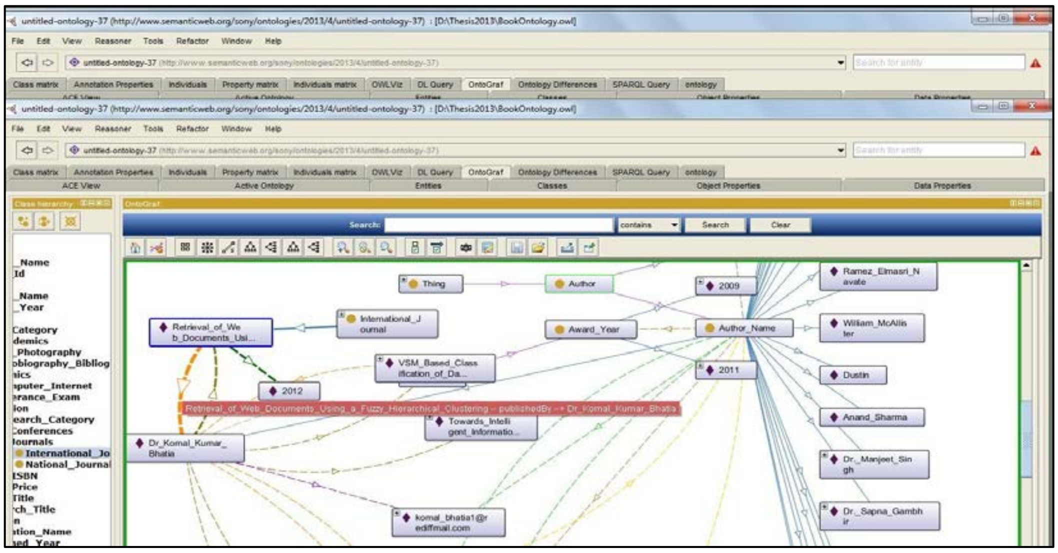

Thing
Author
2009
Author_Name
Award_Year
International_Journal
Retrieval_of_Web_Documents_Usi...
VSM_Based_Classification_of_Da...
2011
2012
Retrieval_of_Web_Documents_Using_a_Fuzzy_Hierarchical_Clustering -- publishedBy --> Dr_Komal_Kumar_Bhatia
Towards_Intelligent_Informatio...
Dr_Komal_Kumar_Bhatia
komal_bhatia1@rediffmail.com
Ramez_Elmasri_Navate
William_McAllister
Dustin
Anand_Sharma
Dr._Manjeet_Singh
Dr._Sapna_Gambhir
Search:
contains
Search
Clear

Figure 3.3 Ontograph of Book Domain Ontology Developed In Protégé.

This module proposes a novel technique for automatic generation of Ontology from available web pages on WWW**.** The Ontology generated automatically using web pages is more generic and rich as compared to the ontology generated manually using Protégé. Both ontologies are merged and a database of <S,P,O> has been designed and created here. The architecture of this module along with algorithms is described in chapter 5. The subcomponents of ontology builder module are as follows:

i) Form Downloader
ii) Classifier
iii) XML/HTML Anlayser
iv) RDF Analyser
v) OWL Analyser
vi) SPO Annotator.

The form downloader downloads the form interfaces from WWW. The classifier according to the format of the page sends the page to corresponding analyser. Each analyser extracts the information from the form pages to create Subject, Predicate and Object. Analyser further send this extracted information to SPO annotator which after cleaning the data store it in semantic database.

3.2.2. Ontology Based Hidden Web Crawler (OHWC): This is the main component of the system. The (OHWC) crawler [96] extracts the hidden web information from WWW using ontology. The architecture for the same is shown in figure 3.4.

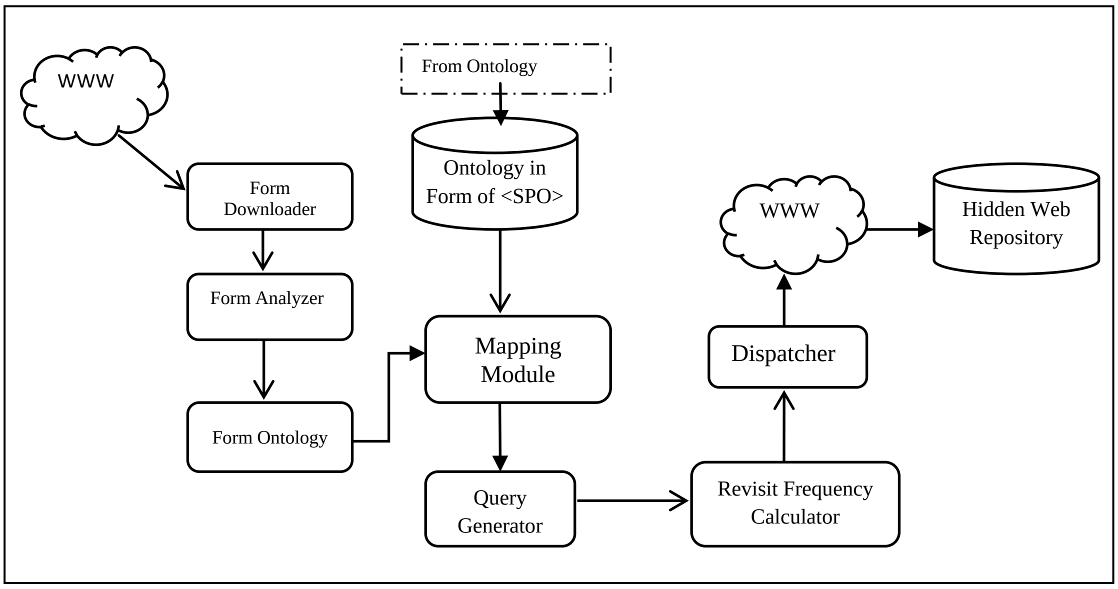


Figure 3.4 Architecture of Ontology based Hidden Web Crawler

The major components of OHWC are:

i) Form Analyzer

ii) Mapping Module

iii) Query Generator

iv) Revisit Frequency Calculator

v) Dispatcher

For making the hidden web crawler work, the form interfaces available on WWW are first downloaded and then analysed. To fill these interfaces with appropriate values; the form ontology is created using the same algorithms defined in chapter 4 for generating ontology. This small ontology is then mapped with the ontology database created before to get the exact values that are needed to fill in those interfaces. So this phase also proposes a novel technique for mapping two ontologies. The component diagram of proposed ontology mapping technique [96] is shown in figure 3.4 below.

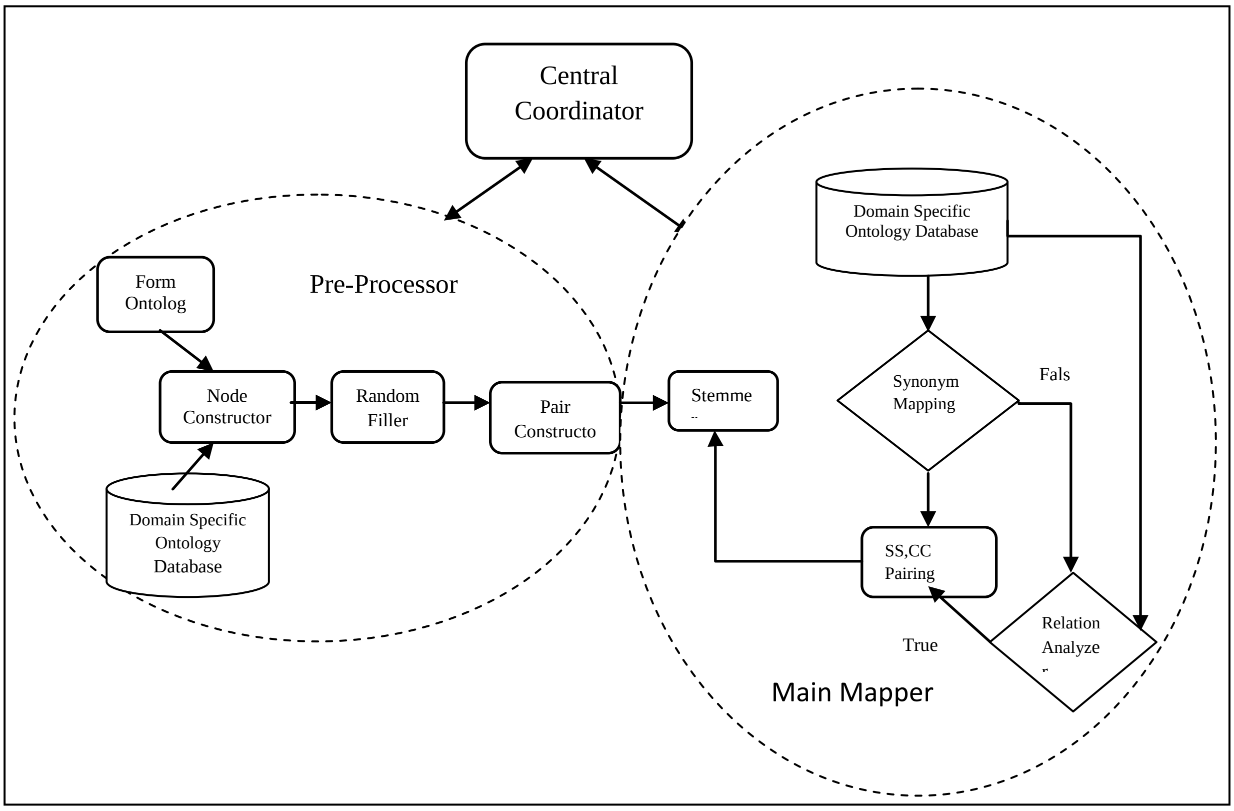


Figure 3.5 Proposed Architecture of Ontology Mapping System.

The technique here maps form interfaces with predefined domain specific ontology database developed by ontology builder. These values once found are used to generate different sets of queries which are fired on WWW and resultant hidden web pages are retrieved.

3.2.3 Ontology Based Indexer: Indexing is a way to represent information from a document so that those conducting searches can retrieve the information easily. Without an index, the search engine would scan every document in the corpus, which would require considerable time and computing power. The pages retrieved by OHWC are stored in a repository; they are needed to be shown to user according to the query fired. To increase the efficiency of the system and to increase the speed of showing result to user, a novel ontology based indexing technique [97] has been proposed. The basic diagram showing indexer module of the system is shown in figure 3.6 below.

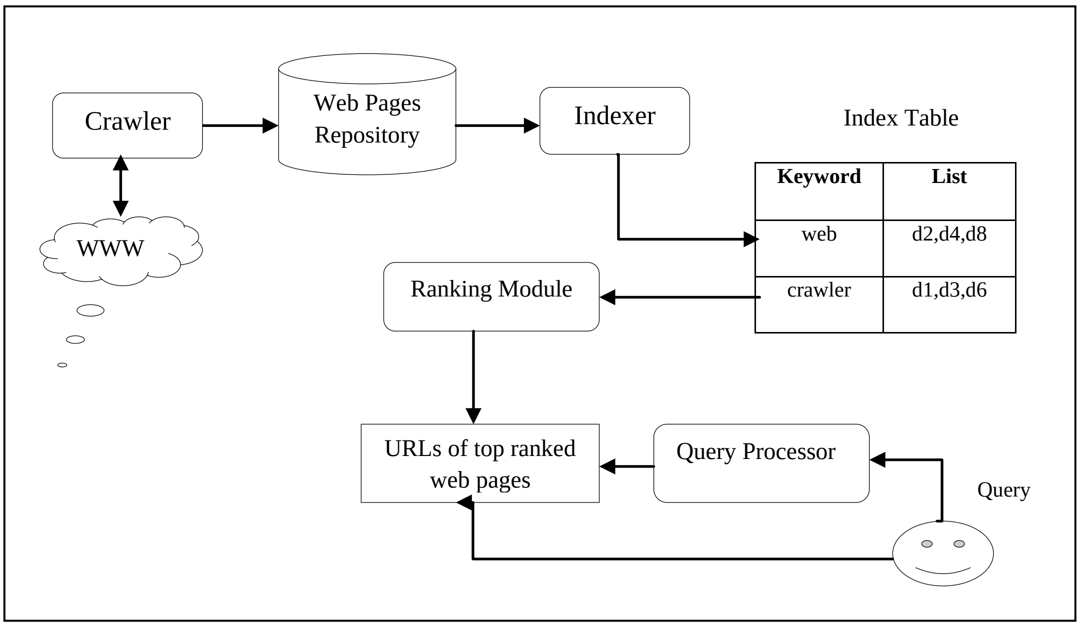


Figure 3.6 Basic Diagram Showing Indexing and Ranking of The System

There are four major components of the proposed ontology based indexing scheme for hidden web namely:

i) Keyword Extractor
ii) Weight Calculator
iii) Concept Annotator
iv) Context Annotator

Ontology based web indexing system attaches different class (concept) to the keyword and does a mapping between keyword and ontology class. But having attached the concept only does not solve the whole problem. Using WordNet the context with every keyword in which it is used is also attached. Due to attachment of context with keyword, more relevant results are retrieved in optimized way in lesser time response to a search query. The final index table which comes after applying the proposed technique is shown in table 3.1 below.

Table 3.1 Table of final index created by proposed ontology based indexing technique.

Keyword	**Concept(class)**	**DocId's**	**Context(Meaning)**
Mouse	Computer	1,4,6,7	Pointing Device
Mouse	Animal	2,3,5,8	Rodent
jaguar	Animal	1,3,4,7,11	Panther
jaguar	Fighter plane	2,6,8,10	Name
jaguar	Car	1,2,4	Brand Name

The proposed model takes the hidden web pages previously downloaded by OHWC to create index. In first step the web pages are indexed word by word. In this step, normal keyword based indexing is used. This results in a cleaned up wordlist for each document, which is stored in a table. Now the keyword to be indexed is mapped to a concepts present in the Ontology.

Each word is now stored with the best matching concept. There are words that map to more than one concepts of ontology which are termed as ambiguous words. Those ambiguous words are analysed and processed further to attach the context in which the word is used. Ontology is updated on regular basis by adding new terms and removing terms stored earlier whenever the pages we are indexing changes and also on regular pages.

3.2.4. Rank Calculator: Page Ranking methods are applied to arrange the results in order of their relevance and importance. These ranking methods can be query dependent or query independent methods. Query-dependent are all ranking methods that are specific to a given query such as word documents frequency, the position of the query terms within the result page, while query-independent factors are attached to the results, regardless of a given query like page content popularity, updated information(in form of last update date/time) etc.

A novel ranking technique for hidden web is proposed here [98] uses factors for both query dependent and query independent ranking methods. It works when user enters queries and the system returns the result according to the indexed data saved as above. Factors of query dependent such as page frequency , query – page content matching are used and factors of query independent such as page content popularity , page source rank, user feedback are also used to design a novel and efficient ranking technique.

The proposed ranking technique comprises of following components:

i) Weight Calculator
ii) Frequency Calculator
iii) Rank Assigner

Brief description of each component is as follows:

i) Weight Calculator (W): In this module weight W is assigned to each url/web page on basis of three factors. One is rating on the web page (W_1), second Users Feedback present on the web page (W_2) and third is Query-Page Content Matching (W_3). These factors are important in hidden web for example in book domain the rating of a book and user’s feedback for a book plays important role in finding an important book. The URL having good rating should come above in the list and same is true for user’s feedback.

ii) Frequency Calculation Module: As the hidden web crawler hidden web pages by filling HTML form on the various websites. It is possible that hidden web crawler fill same set of values in HTML form on more than one website. It may result in generation of web pages having same content from two different websites. Such as two websites of book domain may generate two web pages with different url having description of same book. Frequency calculator will calculate such number of pages and set this as frequency of web page. This frequency is taken in to account to calculate rank as the pages having higher frequency are given preferences hence given higher ranks.

iii) Rank Assigner: In this module final rank value is assigned to each URL. Rank value is used to arrange all URL’s in a ranked list. This rank value, Rv is calculated on basis of weight assigned to each URL (W), and frequency of the web page (f).

Ontology based information retrieval system is designed and implemented for two major domains i) BOOK and ii) Airlines. The implementation and results are discussed and analysed in last chapters.

Chapter 4

GENERATION OF DOMAIN SPECIFIC ONTOLOGY FOR HIDDEN WEB RETRIEVAL USING PROTÉGÉ

4.1 INTRODUCTION

Hidden web is part of WWW that is not part of surface web. It is came in to notice that the Hidden web contain large amount of high quality information that are hidden behind search forms. Majority of hidden web repositories contain files with format pdf, flash, streaming media which contains most real time data and dynamically generated web pages. This kind of information cannot be retrieved by simply traversing the web using keyword searching.

To retrieve those hidden web pages user fill manually in various fields of form pages manually. To automatically extract all this hidden information the crawler (hidden web crawler) must be so intelligent that it understands the interface and fill the required information accurately. This process of understanding and filling the forms automatically can only be efficiently done with the help of ontology. For this purpose, it is required to generate domain specific ontology.

In this chapter detailed description of generating book domain ontology using Protégé [31] has been given and later a novel technique for automatic generation of ontology has been proposed, designed and implemented. The chapter begins with the introduction of idea why book domain has been chosen to generate ontology, and then the hierarchal model used to create knowledge base for the generated ontology is introduced.

4.2 GENERATING DOMAIN SPECIFIC ONTOLOGY USING PROTÉGÉ FOR BOOK DOMAIN

This research work focus on generating the domain specific ontology for retrieving hidden web content. Digital library is one of the domains that contains hidden web. Analysis shows 60 to 70 percent of hidden web information can full fill day to day required information [21]. So it will be useful for users if one can create common search interface through which all online available books can be accessed, leading to generation of book domain ontology. In day to day life a user deals with the books for gathering information or to be in touch with latest news technology and research information. The proposed work try to create proper knowledge base using book domain ontology instead of relational database as to alleviate data redundancy, allow sharing and reuse of information of information. Ontology also gives common understanding of domain structure and easy reasoning in terms of subject, predicate and object (<S, P, O>) triples that are not possible with relational database.

4.3 ONTOLOGICAL MODEL FOR BOOK DOMAIN

Book domain contains the collection of books that are related to Author, Publication, Award etc. For proposing the idea four sub-domains have been identified to generate book domain ontology that are Author, Award, Publication and Book.

4.3.1 Defining Author class: Author is the entity that originates or writes books of his interest. Two fields of Author as shown in figure 4.1 are related to book: i) Author _Name: The name of the author who has written this book/paper and ii) Author_Email Id that may describe contact point of the author.

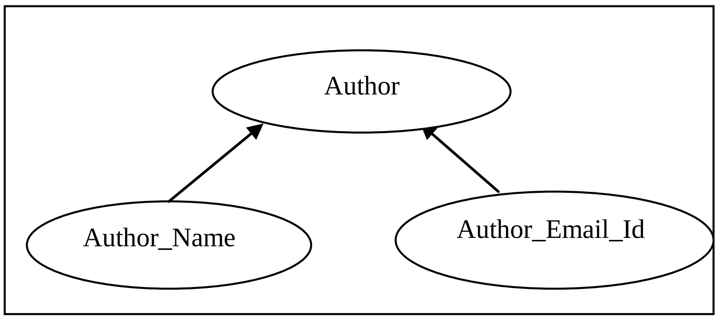


Figure 4.1 Author Sub-domain

4.3.2 Defining Award class: Award is materialistic thing that may be given to any book or any author for his book/paper to recognise its excellence in particular field. Here Award is related to: i) Award Name and ii) Award Year as shown in figure 4.2. Award name is the name associated with award and the Award_year is the year in which it has been given to the author for particular book/paper.

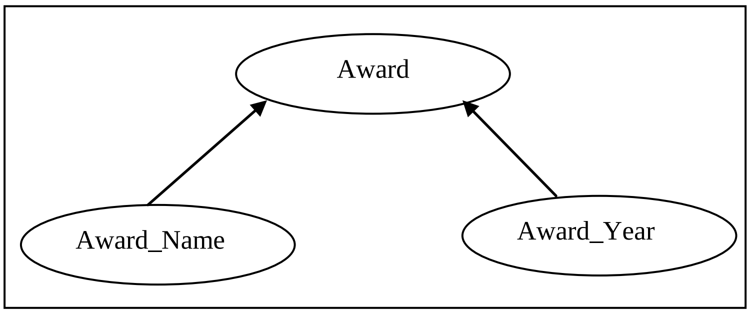


Figure 4.2 Award Sub-domain

4.3.3 Defining Publication class: Publication is an Authority that makes content available to general public by publishing it. The preparation and issuing of a book, journal, piece of music, or other work for public sale is done by publication only as shown in figure 4.3. It is associated with two properties further: i) Publication _Name is the name of the publisher who has published the book and ii) Publication_Year when represents the timestamp date/year of publishing.

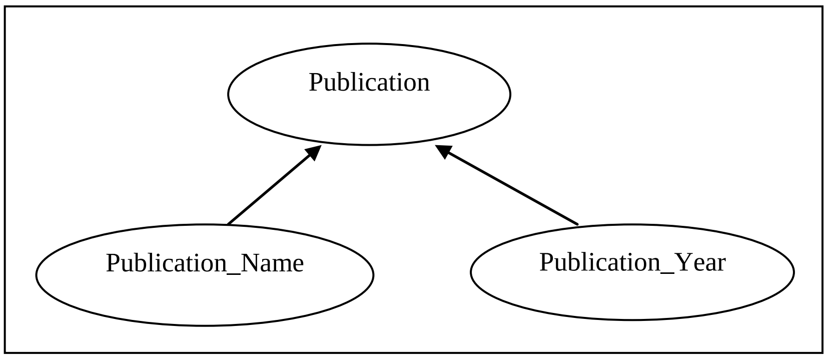


Figure 4.3 Publication Sub-domain

4.3.4 Defining Book class: Book is the main entity to which all above entities are related. It refers to source of information or work of literature. Here book is identified by various properties(fields) like i) Book_Title that represents its unique name, ii) Book_Price that provide information that at what amount it is available to user of interest, iii) Book_category that details about the type of book and iv) Book_ISBN that defines International standard Book number described in figure 4.4. Book category may contain vast varieties of books so proper organisation of books under book category is required. Here it is divided into Academics, Sports, Research category, Computer , Internet, story, Novels etc.

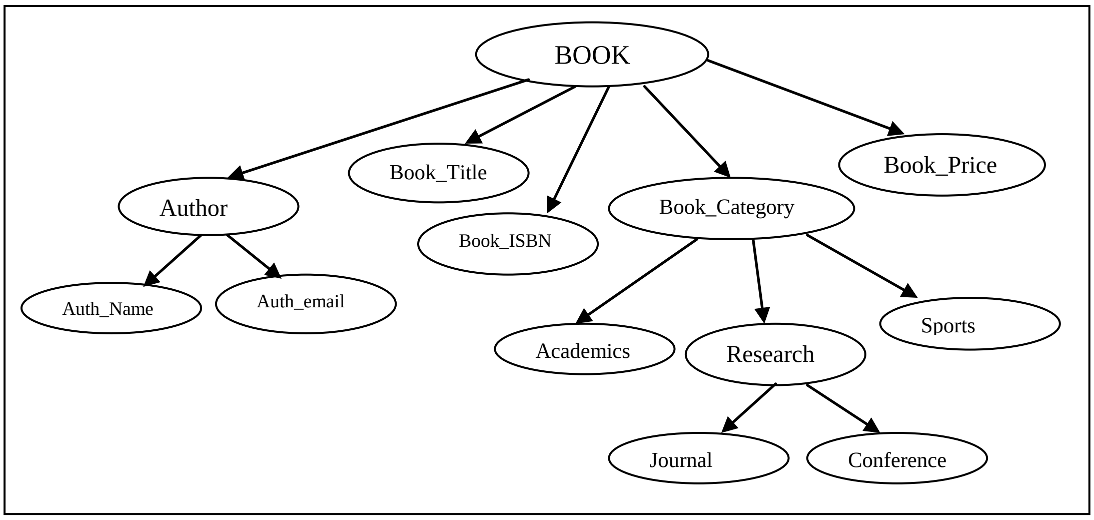


Figure 4.4 Class-Sublass Hierarchy

4.4 PROTÉGÉ ONTOLOGY TOOL

Protégé is a free, open source ontology editor and knowledge-base framework. It is based on Java, is extensible, and provides a plug-and-play environment that makes it is flexible base for application development that also provide for web services. In this work we use Protégé 4.2 latest version of s/w for desktop environment has been used to develop knowledge base for book domain.

4.4.1 Domain Ontology Construction Steps: Using Protege 4.2

Basic steps to create book domain ontology in protégé:

Step 1: Start Protégé 4.2.

Step 2: When Editor Opens either click File →New to create new empty ontology or load the existing ontology by browsing its path.

Step 3: After opening the OWL file its ACE view appears in the editor that uses Attempto Controlled English (ACE) in order to create, view and edit ontology. It display rule set in English language**.**

Step 4: Define Class Hierarchy: There are two possible approaches in developing a class hierarchy. A top-down development process starts with the definition of the most general concepts in the domain and subsequent specialization of the concepts. A bottom-up development process starts with the definition of the most specific classes, the leaves of the hierarchy, with subsequent grouping of these classes into more general concepts.

Step 5: Define Class Attributes/Instances which contributes to data entry to knowledge base. Defining an individual instance of a class requires choosing a class, creating an individual instance of that class, and filling in the slot values

Step 6: Define object Property which specifies the relationship among class-sublcass and class-attributes. Here domain and range is also specified for particular relation.

Step 7: Creation of property matrix: After object properties characteristics of object properties like it may be functional, inverse functional, reflexive, symmetric, anti-symmetric or transitive are defined.

Step 8: Define data property: For specifying data type associated with different objects data property is used which can be defined under data properties tab.

Description with corresponding illustration of each of these steps is given in subsequent sections below.

Step 1 and Step 2: Opening Protege

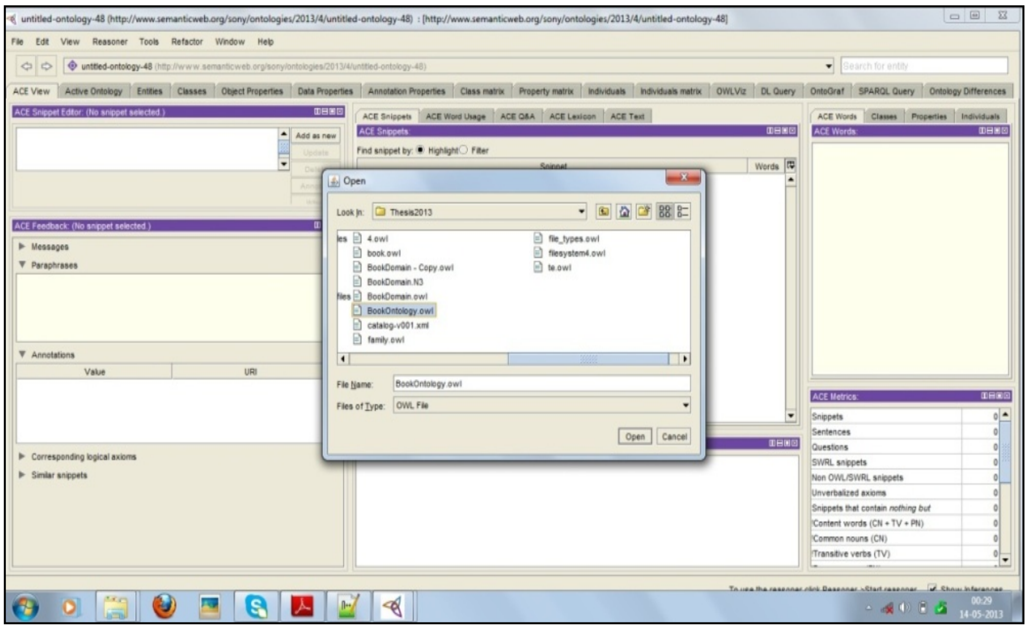
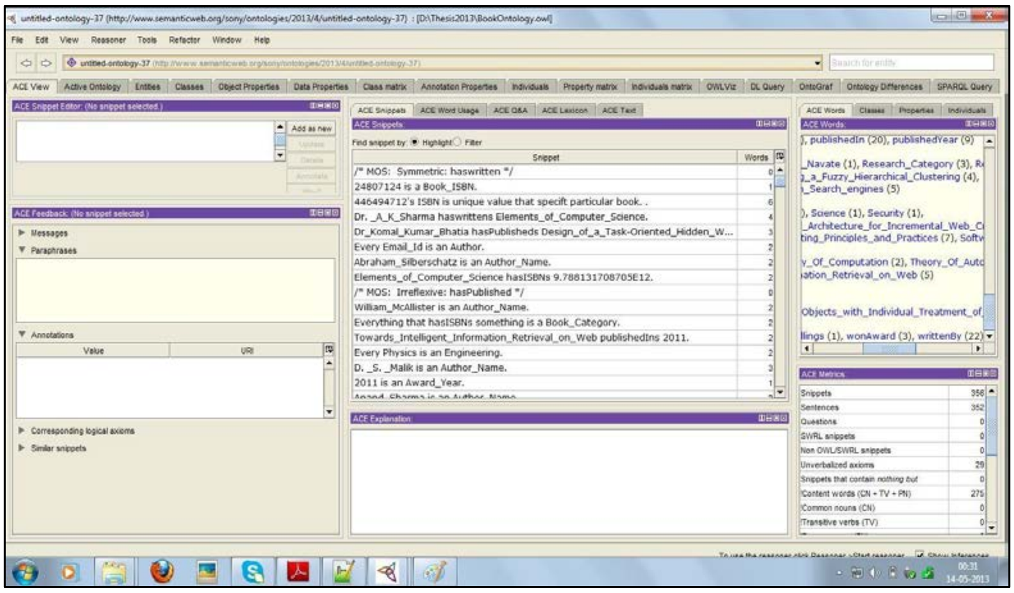


Figure 4.5 Snapshot of Opening a File in Protégé

Step 3: ACE view of editor: Ace view is used to create, view and edit ontology. It display rule set in English language describing the important things ontology is made up of.

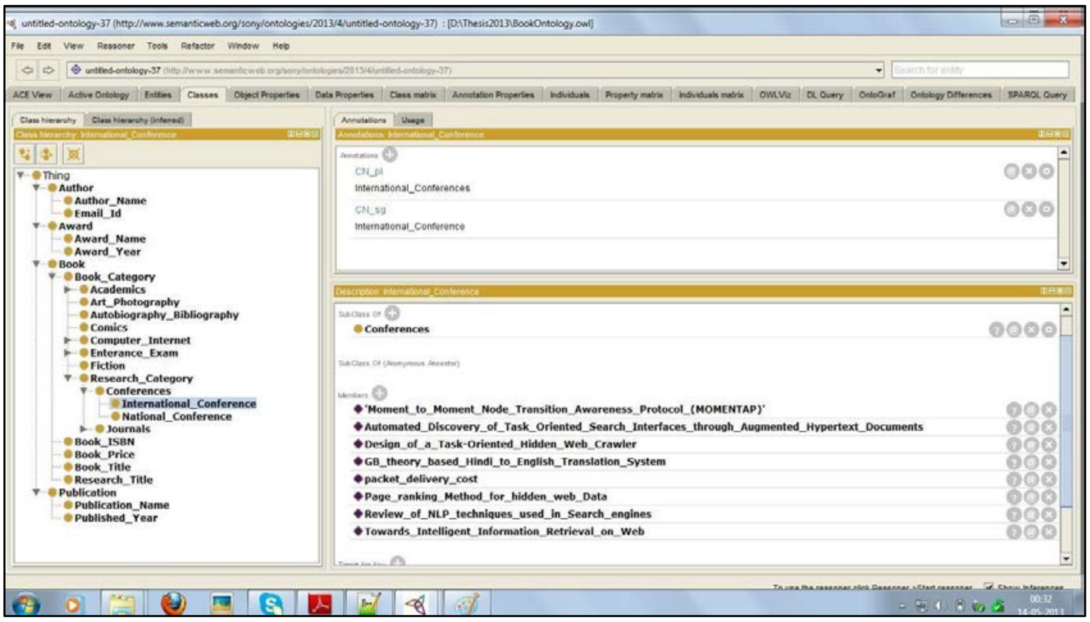


Snippet	Words
/* MOS: Symmetric: haswritten */	0
24807124 is a Book_ISBN.	1
446494712's ISBN is unique value that specift particular book. .	6
Dr. _A_K_Sharma haswrittens Elements_of_Computer_Science.	4
Dr_Komal_Kumar_Bhatia hasPublisheds Design_of_a_Task-Oriented_Hidden_W...	3
Every Email_Id is an Author.	2
Abraham_Silberschatz is an Author_Name.	2
Elements_of_Computer_Science hasISBNs 9.788131708705E12.	2
/* MOS: Irreflexive: hasPublished */	0
William_McAllister is an Author_Name.	2
Everything that hasISBNs something is a Book_Category.	2
Towards_Intelligent_Information_Retrieval_on_Web publishedIns 2011.	2
Every Physics is an Engineering.	2
D. _S. _Malik is an Author_Name.	3
2011 is an Award_Year.	1

ACE Metrics	
Snippets	358
Sentences	352
Questions	0
SWRL snippets	0
Non OWL/SWRL snippets	0
Unverbalized axioms	29
Snippets that contain nothing but	0
Content words (CN + TV + PN)	275
Common nouns (CN)	0
Transitive verbs (TV)	0

Figure 4.6 Snapshot of ACE View of Editor

Step 4: Defining Classes Hierarchy: All the classes along with their corresponding hierarchies are described first. Then super class and sub class relationships are defined. Here Thing is the super class of all concepts defined for particular domain. In the proposed work, everything comes under Thing class. The classes Author, Book, Publications are the superclasses. The Author_name ,Email_id are the sublasses of Author as defined in Figure 4.7. Further Book is the super class having book_category, ISBN,Price and Title as subclasses. Similarly Publication is super class and publication_name and publication_year are the subclasses.



```
Thing
  Author
    Author_Name
    Email_Id
  Award
    Award_Name
    Award_Year
  Book
    Book_Category
      Academics
      Art_Photography
      Autobiography_Bibliography
      Comics
      Computer_Internet
      Enterence_Exam
      Fiction
      Research_Category
        Conferences
          International_Conference
          National_Conference
        Journals
    Book_ISBN
    Book_Price
    Book_Title
    Research_Title
  Publication
    Publication_Name
    Published_Year
```

Annotations: International_Conference

CN_pl
International_Conferences

CN_sg
International_Conference

Description: International_Conference

SubClass Of
Conferences

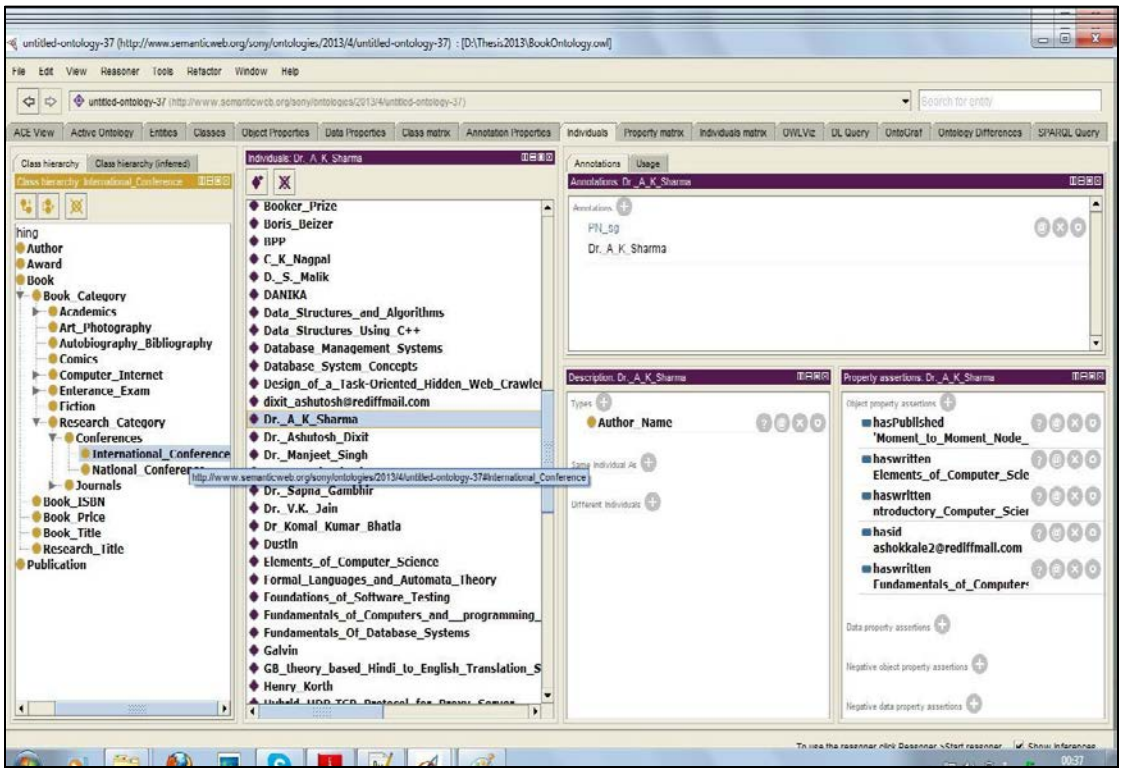
Members
- 'Moment_to_Moment_Node_Transition_Awareness_Protocol_(MOMENTAP)'
- Automated_Discovery_of_Task_Oriented_Search_Interfaces_through_Augmented_Hypertext_Documents
- Design_of_a_Task-Oriented_Hidden_Web_Crawler
- GB_theory_based_Hindi_to_English_Translation_System
- packet_delivery_cost
- Page_ranking_Method_for_hidden_web_Data
- Review_of_NLP_techniques_used_in_Search_engines
- Towards_Intelligent_Information_Retrieval_on_Web

Figure 4.7 Snapshot of Classes Hierarchy

Step 5: Creating Individual view: After Class Hierarchy individuals/attributes are assigned to particular concept (class) which contributes to data in knowledge base. Individuals are the instances of concepts in domain ontology. These instances have particular property attached with concept for which they have been defined. Attributes are the properties associated between two classes or a class and a subclass. For each attribute one class is Subject and the other is Object. Each class and subclass have individuals and attributes associated to them. Individual view shows all the classes, subclasses and corresponding properties associated with each instance. As an example figure 4.8 below describes the class and subclass hierarchies for the instance Dr._A_K_Sharma as Author's instance.

i) It is instance of Author_Name.
ii) Related with object property hasPublished , hasWritten and hasId.

The author having Author_name is associated with has_written and has_published properties. Here for each instance values are stored.



untitled-ontology-37 (http://www.semanticweb.org/sony/ontologies/2013/4/untitled-ontology-37) : [D:\Thesis2013\BookOntology.owl]

untitled-ontology-37 (http://www.semanticweb.org/sony/ontologies/2013/4/untitled-ontology-37)

Class hierarchy: International_Conference

hing
- Author
- Award
- Book
  - Book_Category
    - Academics
    - Art_Photography
    - Autobiography_Bibliography
    - Comics
    - Computer_Internet
    - Enterance_Exam
    - Fiction
    - Research_Category
      - Conferences
        - International_Conference
        - National_Confere
      - Journals
  - Book_ISBN
  - Book_Price
  - Book_Title
  - Research_Title
- Publication

http://www.semanticweb.org/sony/ontologies/2013/4/untitled-ontology-37#International_Conference

Individuals: Dr._A_K_Sharma

- Booker_Prize
- Boris_Beizer
- BPP
- C_K_Nagpal
- D._S._Malik
- DANIKA
- Data_Structures_and_Algorithms
- Data_Structures_Using_C++
- Database_Management_Systems
- Database_System_Concepts
- Design_of_a_Task-Oriented_Hidden_Web_Crawler
- dixit_ashutosh@rediffmail.com
- Dr._A_K_Sharma
- Dr._Ashutosh_Dixit
- Dr._Manjeet_Singh
- Dr._Sapna_Gambhir
- Dr._V.K._Jain
- Dr_Komal_Kumar_Bhatia
- Dustin
- Elements_of_Computer_Science
- Formal_Languages_and_Automata_Theory
- Foundations_of_Software_Testing
- Fundamentals_of_Computers_and__programming_
- Fundamentals_Of_Database_Systems
- Galvin
- GB_theory_based_Hindi_to_English_Translation_S
- Henry_Korth

Annotations: Dr._A_K_Sharma

PN_sg
Dr._A_K_Sharma

Description: Dr._A_K_Sharma

Types
Author_Name

Property assertions: Dr._A_K_Sharma

Object property assertions
- hasPublished 'Moment_to_Moment_Node_
- haswritten Elements_of_Computer_Scie
- haswritten ntroductory_Computer_Scie
- hasid ashokkale2@rediffmail.com
- haswritten Fundamentals_of_Computers

To use the reasoner click Reasoner->Start reasoner

00:37

Figure 4.8 Snapshot of Individual view

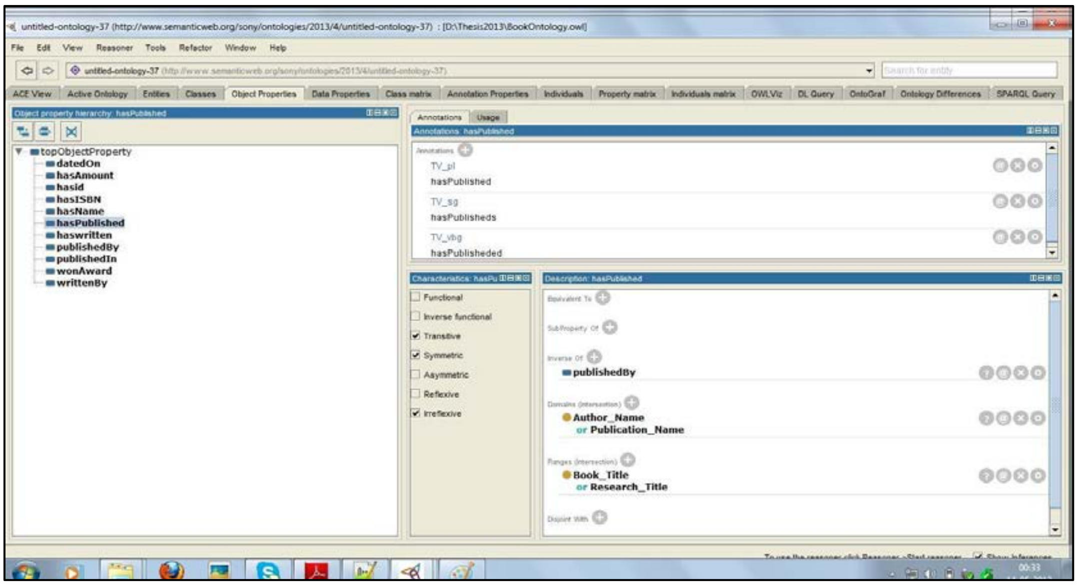
Step 6: Defining Object Property: Property states the relationship between two entities, it is also known as predicate. The relation specified in ontology is in the form

of <S, P, O > triple. Here subject and object are entity from concepts set and predicate links the subject and objects. The instances defined above contribute to Subject, attributes contribute to Object and properties contribute to Predicate for each tuple.
Book Domain Ontology defined here uses various object properties as defined in table 4.1 and the snapshot is shown in figure 4.9 for the same. Here Domain is defined as collection of subject and range defines collection on object. Objectproperty maps the Domain and Range, as mapping done in functional mapping.

Table 4.1 Objectproperty Table

Object Property	**Subject (Domain)**	**Object (Range)**
datedOn	Author_Name	Author_Year
hasAmount	Book_Title	Book_Price
hasId	Author_Name	Author_EmailID
hasISBN	Book_Title Book_Ctaegory	Book_ISBN
hasName	Book_Category	Book_Title
hasPublished	Author_Name or Publication_Name	Research_Title or Book_Title
hasWritten	Author_Name	Book_Title
publishedBy	Book_Title Research_Title	Publication_Name
publishedIn	Book_Title Research_Title	Published_Year
wonAward	Book_Title Author_Name	Award_Name
writtenBy	Book_title	Author_Name

For each particular property database have more than one entries of subject and object. At one time for one predicate one instance may behave as Subject and for the other predicate value it may behave as Object in another tuple.


untitled-ontology-37 (http://www.semanticweb.org/sony/ontologies/2013/4/untitled-ontology-37) : [D:\Thesis2013\BookOntology.owl]

Object property hierarchy: hasPublished

- topObjectProperty
  - datedOn
  - hasAmount
  - hasid
  - hasISBN
  - hasName
  - hasPublished
  - haswritten
  - publishedBy
  - publishedIn
  - wonAward
  - writtenBy

Annotations: hasPublished

- TV_pl: hasPublished
- TV_sg: hasPublisheds
- TV_vbg: hasPublisheded

Characteristics: hasPu

- [ ] Functional
- [ ] Inverse functional
- [x] Transitive
- [x] Symmetric
- [ ] Asymmetric
- [ ] Reflexive
- [x] Irreflexive

Description: hasPublished

Equivalent To

SubProperty Of

Inverse Of: publishedBy

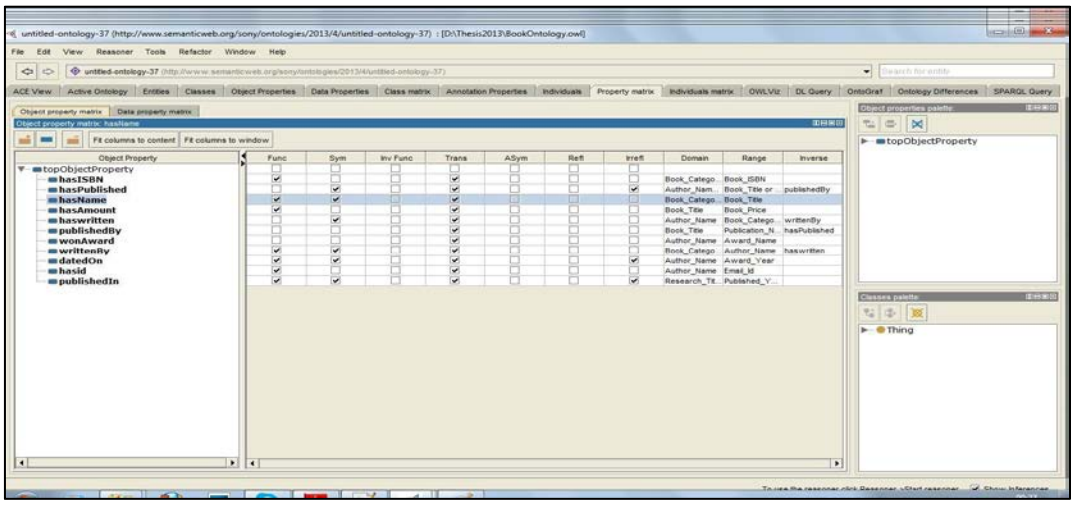
Domains (intersection): Author_Name or Publication_Name

Ranges (intersection): Book_Title or Research_Title

Disjoint With

Figure 4.9 Snapshot Showing Object Properties

Step 7: Creating Property matrix: After object properties characteristics of object properties are defined. These may be functional, inverse functional, reflexive, symmetric, anti-symmetric or transitive figure 4.10. These properties are defined under property matrix tab. For Example writtenBy is inverse of hasWritten and have inverse functional relationship.


Object property matrix: hasName

Object Property	Func	Sym	Inv Func	Trans	ASym	Refl	Irrefl	Domain	Range	Inverse
topObjectProperty										
hasISBN	✓			✓				Book_Catego...	Book_ISBN	
hasPublished		✓		✓			✓	Author_Nam...	Book_Title or ...	publishedBy
hasName	✓	✓		✓				Book_Catego...	Book_Title	
hasAmount	✓			✓				Book_Title	Book_Price	
haswritten		✓		✓				Author_Name	Book_Catego...	writtenBy
publishedBy				✓				Book_Title	Publication_N...	hasPublished
wonAward				✓				Author_Name	Award_Name	
writtenBy	✓	✓		✓				Book_Catego...	Author_Name	haswritten
datedOn	✓	✓		✓			✓	Author_Name	Award_Year	
hasid	✓			✓				Author_Name	Email_Id	
publishedIn	✓	✓		✓			✓	Research_Tit...	Published_Y...	

Figure 4.10 Snapshot Showing Property Matrix

Step 8: Define Data property: For specifying data type associated with different objects data property is used. Data properties are defined under data properties tab. Data property specifies the range of instance associated with individual concept. For example Name has range String, Amount has range type integer, ISBN has alphanumeric value shown in figure 4.11 below.

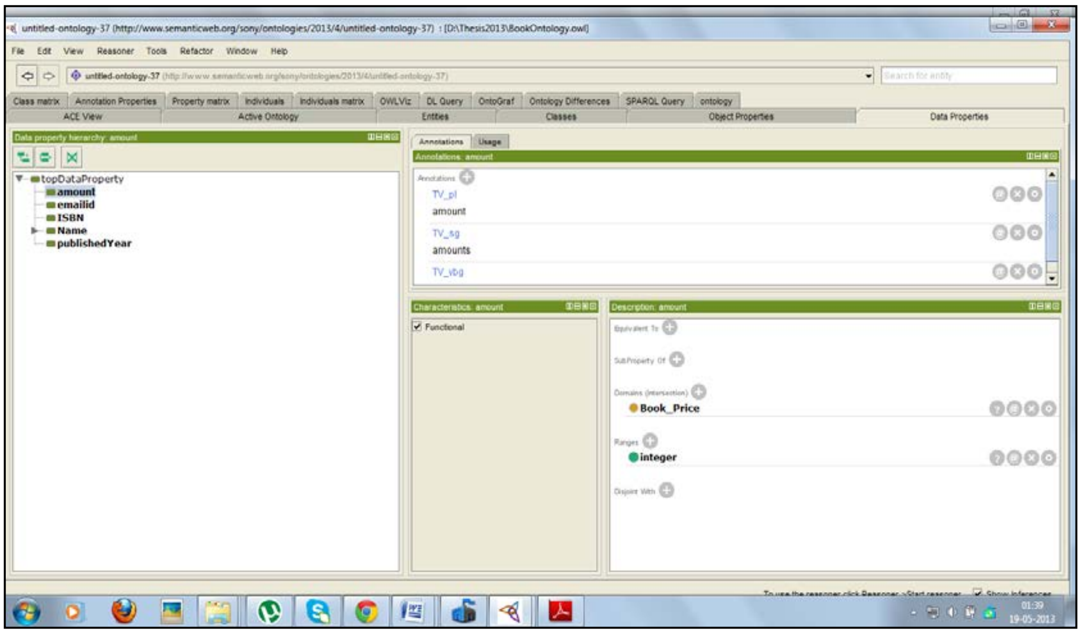


Figure 4.11 Snapshot Showing Data Property

4.4.2 Viewing Ontology

There are two methods to see the whole graphical view of the ontology including classes, subclasses, instances and relationships among them first is i) OWL Viz and second is ii) Ontograf. These are two different tabs available in Protégé. Using OWL Viz the overall class hierarchy can be seen and by using Ontograf tab for a particular instance one can see all the classes, subclasses and properties the instance is related to.

i) OWL Viz: After completing the ontology one can view the overall OWL class hierarchy view under OWL Viz tab. For viewing the graphical view of the ontology GraphViz need to be installed on the system. OWL Viz basically depicts the “is-a” relationship by showing the class and subclass relationships graphically. The hierarchy view of our domain is depicted in the figure 4.12 .

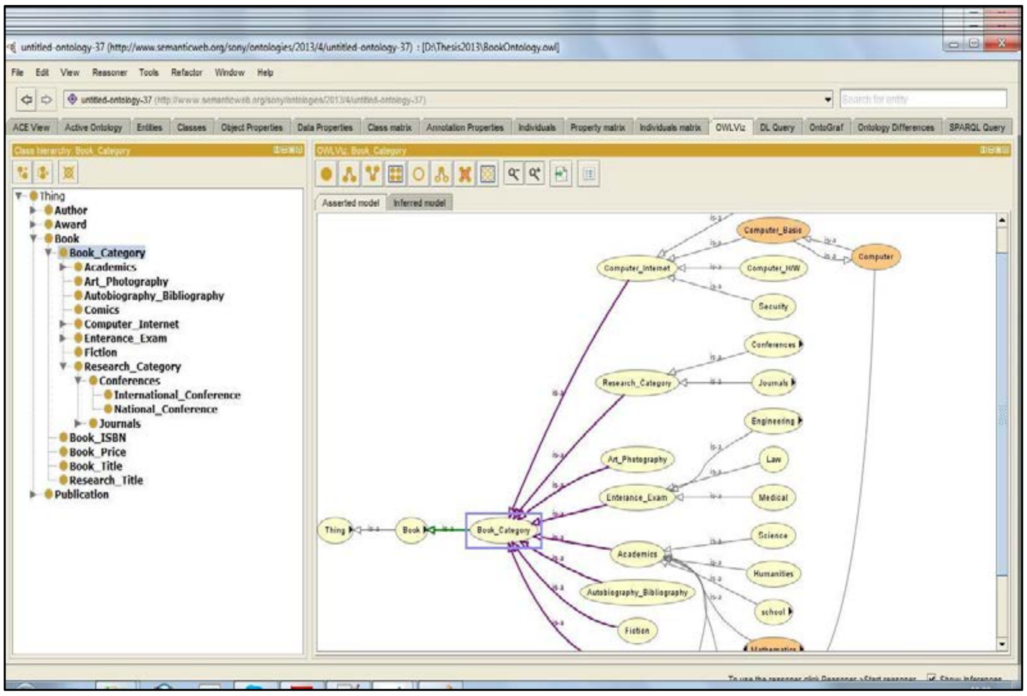


Figure 4.12 Snapshot Showing OWL Class Hierarchy

ii) Ontograf : To provide support for interactively navigating the relationship of OWL created above, one can use ontograf utility that support display of individual, domain, range, property, relationship along with the superclass-subclass relationship,. This provides the familiar view to overall ontology generated as depicted by figure 4.13.

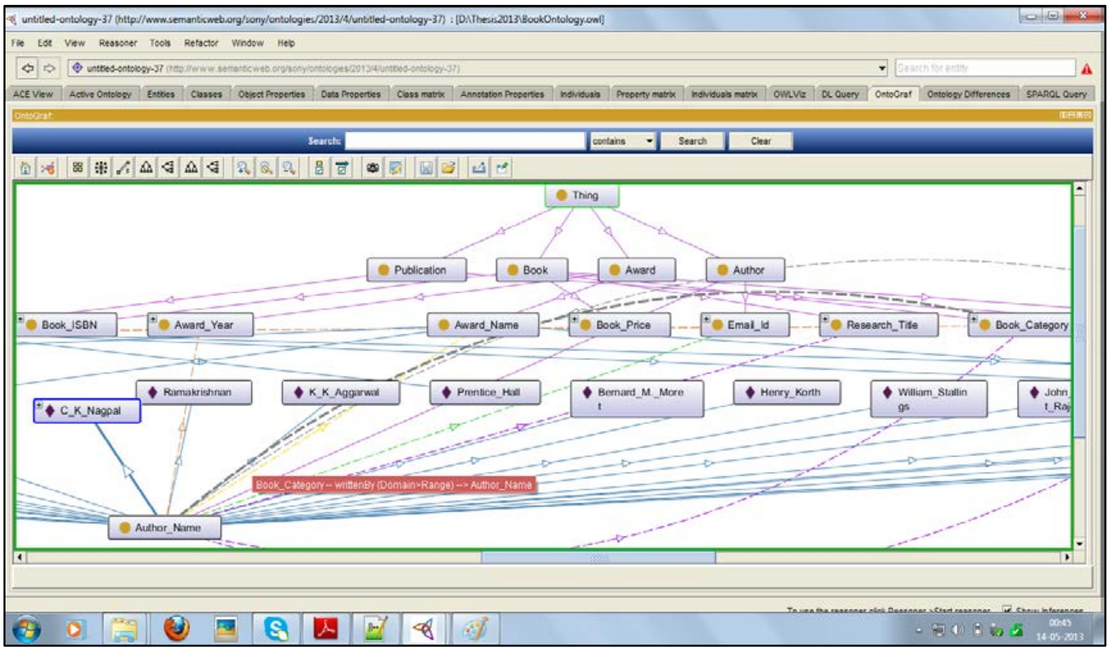


Figure 4.13 Snapshot showing Ontograf.

By clicking on particular individual/instance all the relations with other classes and instances to which it is related can be easily seen in Ontograf. Putting the mouse on any of the arrow shows the name of relation also as shown in figure above where written by relation can be easily seen between author_name and actual name of the author.

4.5 INSTANCE EXAMPLES

The snapshots of various scenarios that depict the proper relationship of any instances to all other property, other instance and concepts have been taken and shown in figures below. In figure 4.14 Dr. A_K_Sharma has been shown as instance showing all its relationship to other concepts along with different properties and their values

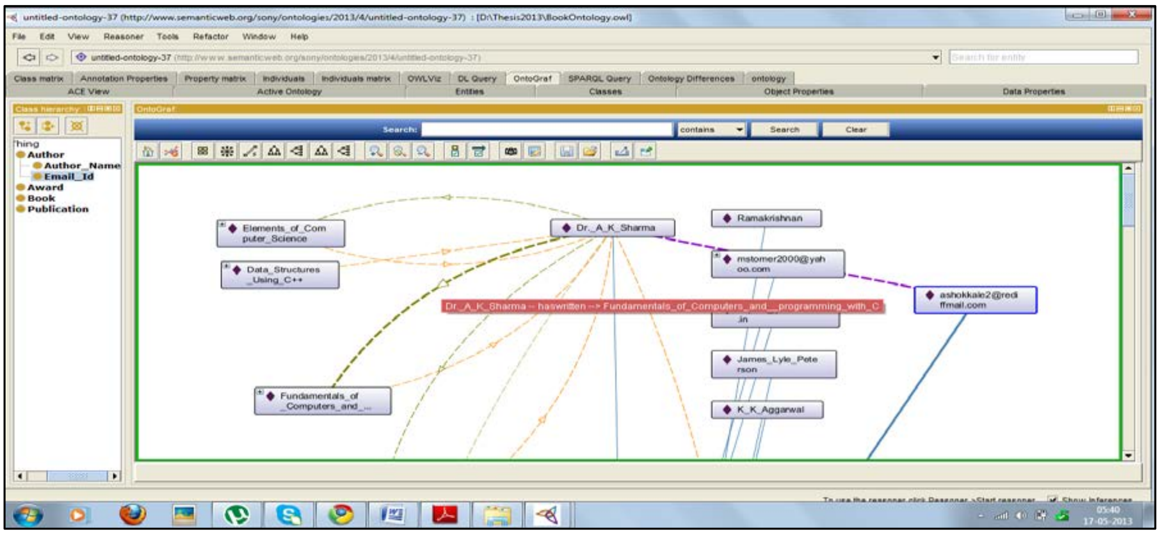

untitled-ontology-37 (http://www.semanticweb.org/sony/ontologies/2013/4/untitled-ontology-37) : [D:\Thesis2013\BookOntology.owl]
File Edit View Reasoner Tools Refactor Window Help
untitled-ontology-37 (http://www.semanticweb.org/sony/ontologies/2013/4/untitled-ontology-37)
Search for entity
Class matrix
Annotation Properties
Property matrix
Individuals
Individuals matrix
OWLViz
DL Query
OntoGraf
SPARQL Query
Ontology Differences
ontology
ACE View
Active Ontology
Entities
Classes
Object Properties
Data Properties
Class hierarchy:
OntoGraf
Search:
contains
Search
Clear
Thing
Author
Author_Name
Email_Id
Award
Book
Publication
Elements_of_Computer_Science
Data_Structures_Using_C++
Fundamentals_of_Computers_and_...
Dr._A_K_Sharma
Ramakrishnan
mstomer2000@yahoo.com
ashokkale2@rediffmail.com
Dr_A_K_Sharma -- haswritten --> Fundamentals_of_Computers_and__programming_with_C
.in
James_Lyle_Peterson
K_K_Aggarwal
To use the reasoner click Reasoner->Start reasoner
Show Inferences
05:40
17-05-2013

Figure 4.14 Ontograf for Instance Dr._A_K_Sharma

After developing the entire book domain ontology the knowledge base is ready to feed as input for reasoning.

Figure 4.15 shows Dr_Komal_Kumar_Bhatia as member of ontology and various relationships with other concepts and also the ontograph shows various properties attached with instances.

Figure 4.16 shows another instance Dr. Ashutosh dixit and all its properties.

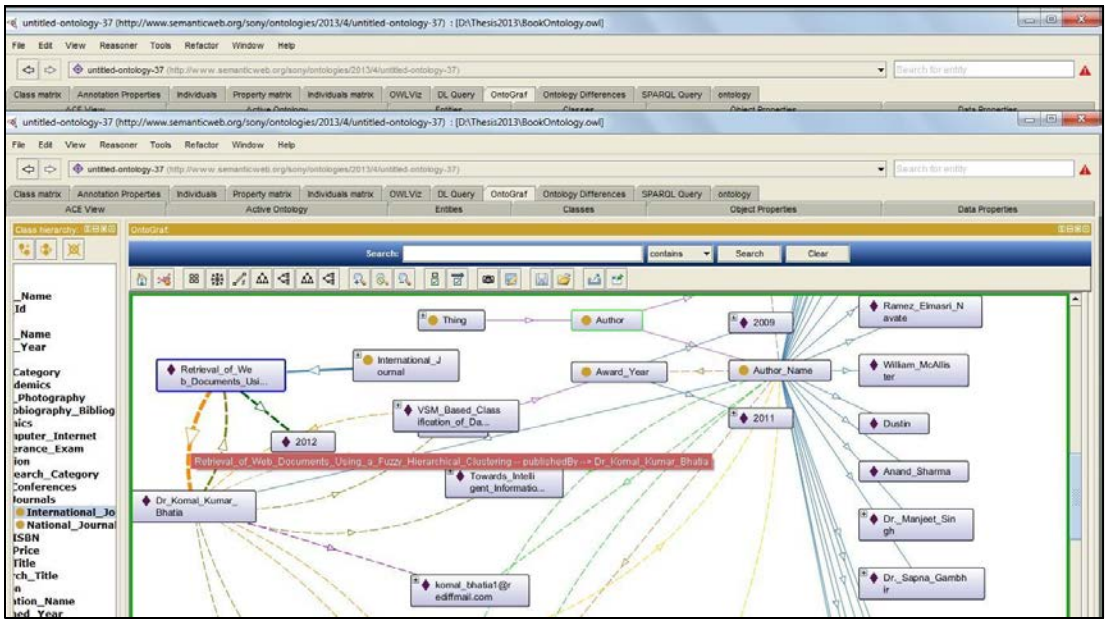


Figure 4.15 Ontograf for Instance Dr._Komal_Kumar_Bhatia

Taking Dr._Ashutosh_Dixit as instance .

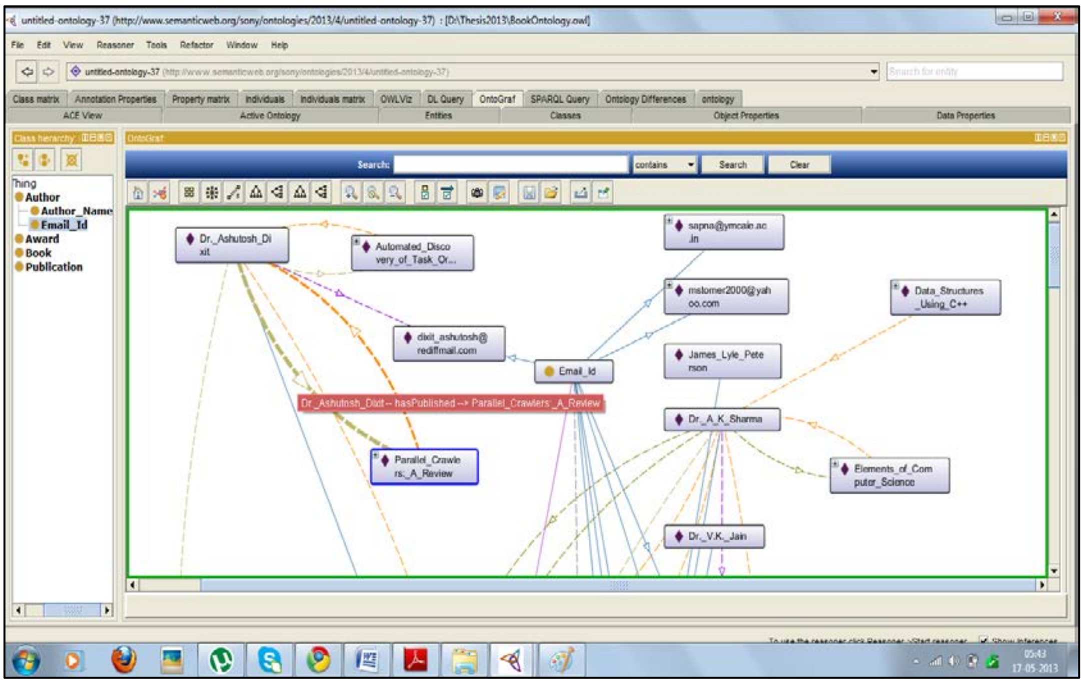


Figure 4.16 Ontograf for Instance Dr._Ashutosh_Dixit

4.6 ONTOLOGY BASED QUERY PROCESSING AND RESULTS

Now the system is having complete knowledge base of book domain for which Ontology has been created which is ready to handle user's book domain related query. For available proposed ontology there is the need for designing of an ontology-based querying system which maps the information asked by the user to the knowledge stored in the ontology. Here in this section discussion on how to create query interface, which query language can be use to answer user's query is done. Also the snapshots of various user queries and their results are described.

4.6.1 Ontology based query system: SPARQL query language

For creating query interface for user Netbeans IDE 6.9.1 and incorporate jena API[100] that has java library is used to help system developer to provide storage view and way to query over knowledge base. The queries that can extract information in RDF triple which can display class hierarchy, display subject object relationship and support SPARQL query over knowledge base have been created.

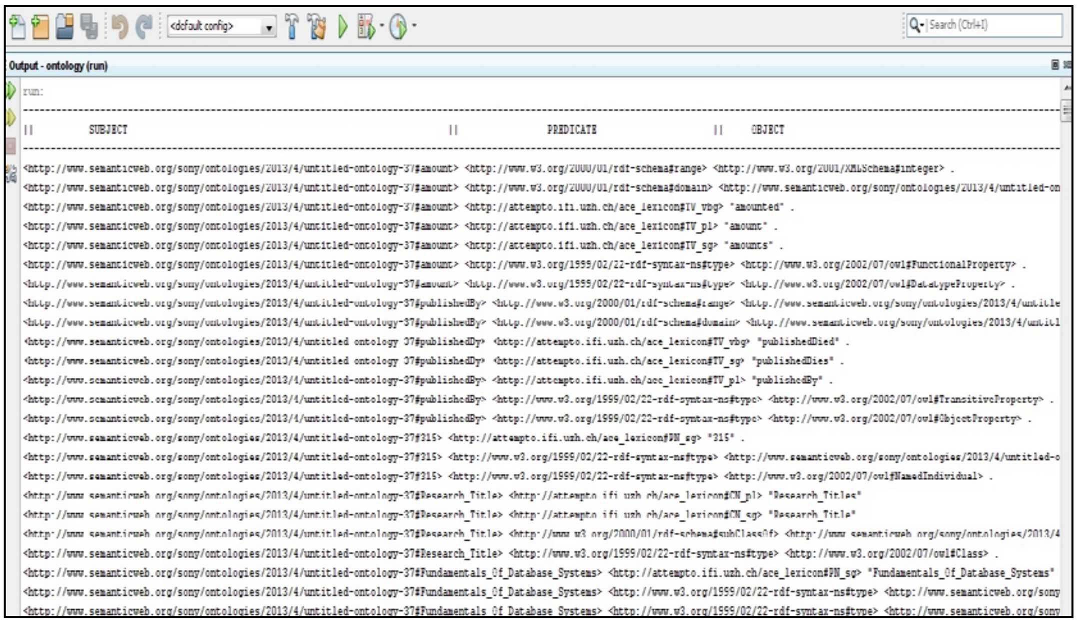
SPARQL is the query language of the Semantic Web. It lets us:

i) Pull values from structured and semi-structured data.
ii) Explore data by querying unknown relationships.
iii) Perform complex joins of disparate databases in a single, simple query.
iv) Transform RDF data from one vocabulary to another.

SPARQL is used to express queries across diverse data sources, whether the data is stored natively as RDF or viewed as RDF. SPARQL contains capabilities for querying required and optional graph patterns. SPARQL also supports extensible value testing and constraining queries by source RDF graph. The results of SPARQL queries can be results sets or RDF graphs.

4.6.2 Query Processing: In this section firstly the figures that show the organisation of knowledge base in RDF triple, class hierarchy are described. Then various user queries in SPARQL format that are given as inputs to system and result obtained by executing those queries are shown in form of snapshots.

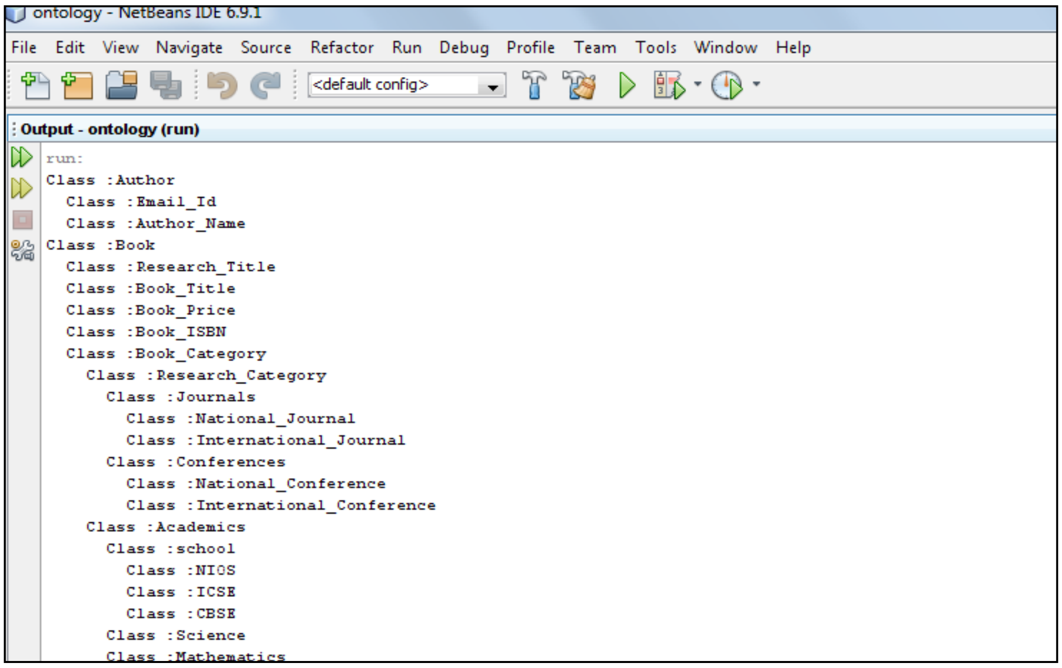
i) Knowledge Base Organisation: Knowledge base is stored in RDF triplets form. Each triple has <S P O> subject predicate and object relationship. The same is shown in figure 4.17 below.



```
run:
----------------------------------------------------------------------------------------------------------------------------------------
||        SUBJECT                                                        ||              PREDICATE                    ||     OBJECT
----------------------------------------------------------------------------------------------------------------------------------------
<http://www.semanticweb.org/sony/ontologies/2013/4/untitled-ontology-37#amount> <http://www.w3.org/2000/01/rdf-schema#range> <http://www.w3.org/2001/XMLSchema#integer> .
<http://www.semanticweb.org/sony/ontologies/2013/4/untitled-ontology-37#amount> <http://www.w3.org/2000/01/rdf-schema#domain> <http://www.semanticweb.org/sony/ontologies/2013/4/untitled-on
<http://www.semanticweb.org/sony/ontologies/2013/4/untitled-ontology-37#amount> <http://attempto.ifi.uzh.ch/ace_lexicon#TV_vbg> "amounted" .
<http://www.semanticweb.org/sony/ontologies/2013/4/untitled-ontology-37#amount> <http://attempto.ifi.uzh.ch/ace_lexicon#TV_pl> "amount" .
<http://www.semanticweb.org/sony/ontologies/2013/4/untitled-ontology-37#amount> <http://attempto.ifi.uzh.ch/ace_lexicon#TV_sg> "amounts" .
<http://www.semanticweb.org/sony/ontologies/2013/4/untitled-ontology-37#amount> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://www.w3.org/2002/07/owl#FunctionalProperty> .
<http://www.semanticweb.org/sony/ontologies/2013/4/untitled-ontology-37#amount> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://www.w3.org/2002/07/owl#DatatypeProperty> .
<http://www.semanticweb.org/sony/ontologies/2013/4/untitled-ontology-37#publishedBy> <http://www.w3.org/2000/01/rdf-schema#range> <http://www.semanticweb.org/sony/ontologies/2013/4/untitle
<http://www.semanticweb.org/sony/ontologies/2013/4/untitled-ontology-37#publishedBy> <http://www.w3.org/2000/01/rdf-schema#domain> <http://www.semanticweb.org/sony/ontologies/2013/4/untitl
<http://www.semanticweb.org/sony/ontologies/2013/4/untitled-ontology-37#publishedBy> <http://attempto.ifi.uzh.ch/ace_lexicon#TV_vbg> "publishedDied" .
<http://www.semanticweb.org/sony/ontologies/2013/4/untitled-ontology-37#publishedBy> <http://attempto.ifi.uzh.ch/ace_lexicon#TV_sg> "publishedDies" .
<http://www.semanticweb.org/sony/ontologies/2013/4/untitled-ontology-37#publishedBy> <http://attempto.ifi.uzh.ch/ace_lexicon#TV_pl> "publishedBy" .
<http://www.semanticweb.org/sony/ontologies/2013/4/untitled-ontology-37#publishedBy> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://www.w3.org/2002/07/owl#TransitiveProperty> .
<http://www.semanticweb.org/sony/ontologies/2013/4/untitled-ontology-37#publishedBy> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://www.w3.org/2002/07/owl#ObjectProperty> .
<http://www.semanticweb.org/sony/ontologies/2013/4/untitled-ontology-37#315> <http://attempto.ifi.uzh.ch/ace_lexicon#PN_sg> "315" .
<http://www.semanticweb.org/sony/ontologies/2013/4/untitled-ontology-37#315> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://www.semanticweb.org/sony/ontologies/2013/4/untitled-o
<http://www.semanticweb.org/sony/ontologies/2013/4/untitled-ontology-37#315> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://www.w3.org/2002/07/owl#NamedIndividual> .
<http://www.semanticweb.org/sony/ontologies/2013/4/untitled-ontology-37#Research_Title> <http://attempto.ifi.uzh.ch/ace_lexicon#CN_pl> "Research_Titles"
<http://www.semanticweb.org/sony/ontologies/2013/4/untitled-ontology-37#Research_Title> <http://attempto.ifi.uzh.ch/ace_lexicon#CN_sg> "Research_Title"
<http://www.semanticweb.org/sony/ontologies/2013/4/untitled-ontology-37#Research_Title> <http://www.w3.org/2000/01/rdf-schema#subClassOf> <http://www.semanticweb.org/sony/ontologies/2013/4
<http://www.semanticweb.org/sony/ontologies/2013/4/untitled-ontology-37#Research_Title> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://www.w3.org/2002/07/owl#Class> .
<http://www.semanticweb.org/sony/ontologies/2013/4/untitled-ontology-37#Fundamentals_Of_Database_Systems> <http://attempto.ifi.uzh.ch/ace_lexicon#PN_sg> "Fundamentals_Of_Database_Systems"
<http://www.semanticweb.org/sony/ontologies/2013/4/untitled-ontology-37#Fundamentals_Of_Database_Systems> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://www.semanticweb.org/sony
<http://www.semanticweb.org/sony/ontologies/2013/4/untitled-ontology-37#Fundamentals_Of_Database_Systems> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://www.semanticweb.org/sony
```

Figure 4.17 Knowledge Base in RDF Triples

Figure 4.18 describes the class hierarchy relationship in NETBEANS IDE using jena API over owl file.


ontology - NetBeans IDE 6.9.1
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
<default config>
Output - ontology (run)

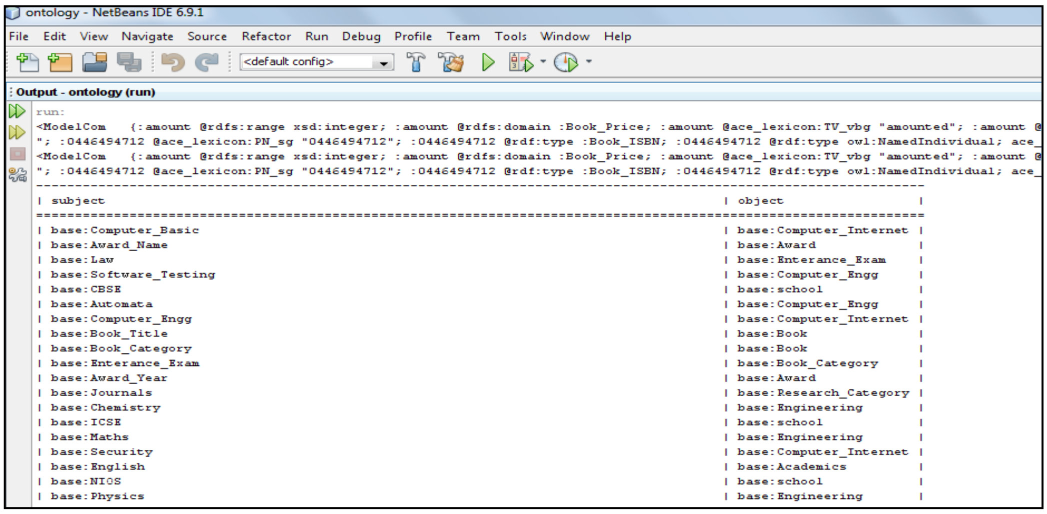
```
run:
Class :Author
  Class :Email_Id
  Class :Author_Name
Class :Book
  Class :Research_Title
  Class :Book_Title
  Class :Book_Price
  Class :Book_ISBN
  Class :Book_Category
    Class :Research_Category
      Class :Journals
        Class :National_Journal
        Class :International_Journal
      Class :Conferences
        Class :National_Conference
        Class :International_Conference
    Class :Academics
      Class :school
        Class :NIOS
        Class :ICSE
        Class :CBSE
      Class :Science
      Class :Mathematics
```

Figure 4.18 Class Hierarchy of Book Domain OWL

ii) User Query and Result: In this subsection the snapshots showing the queries entered and the corresponding result retrieved. If user wants to the list of all subjects and objects which are related to each other? The query entered is shown as input and the snapshot is shown as output for each query. The query shown here is taken in same format as was written in code to get the desired result. The result is shown in figure 4.19.

a) Input Query: "*SELECT ?subject ?object WHERE { ?subject rdfs:subClassOf ?object }*"

Output:

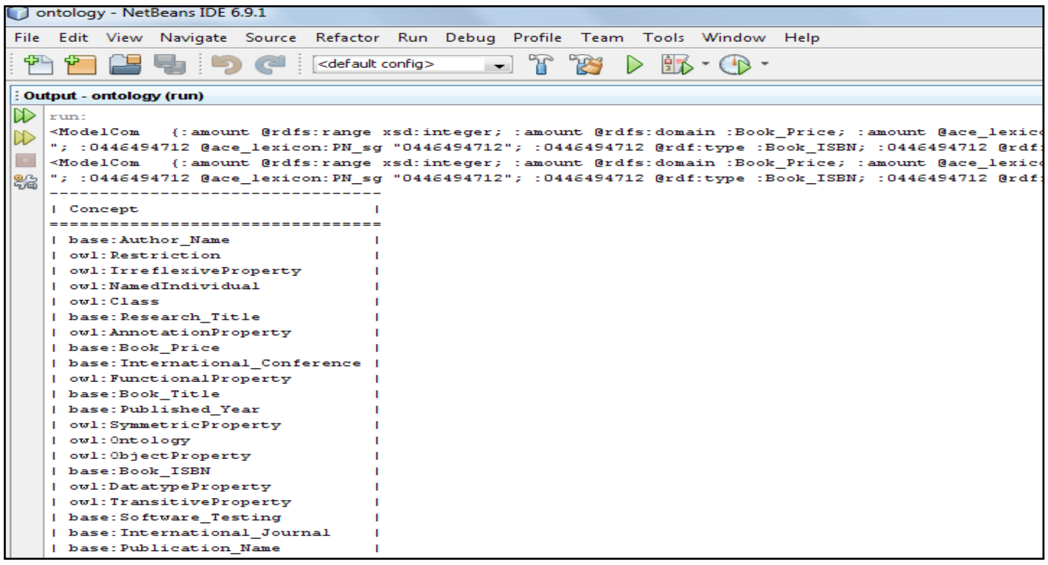


ontology - NetBeans IDE 6.9.1
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
<default config>
Output - ontology (run)

```
run:
<ModelCom   {:amount @rdfs:range xsd:integer; :amount @rdfs:domain :Book_Price; :amount @ace_lexicon:TV_vbg "amounted"; :amount @
"; :0446494712 @ace_lexicon:PN_sg "0446494712"; :0446494712 @rdf:type :Book_ISBN; :0446494712 @rdf:type owl:NamedIndividual; ace_
<ModelCom   {:amount @rdfs:range xsd:integer; :amount @rdfs:domain :Book_Price; :amount @ace_lexicon:TV_vbg "amounted"; :amount @
"; :0446494712 @ace_lexicon:PN_sg "0446494712"; :0446494712 @rdf:type :Book_ISBN; :0446494712 @rdf:type owl:NamedIndividual; ace_
----------------------------------------------------------------------------------------------------------------------
| subject                                                                                     | object                   |
======================================================================================================================
| base:Computer_Basic                                                                         | base:Computer_Internet |
| base:Award_Name                                                                             | base:Award             |
| base:Law                                                                                    | base:Enterance_Exam    |
| base:Software_Testing                                                                       | base:Computer_Engg     |
| base:CBSE                                                                                   | base:school            |
| base:Automata                                                                               | base:Computer_Engg     |
| base:Computer_Engg                                                                          | base:Computer_Internet |
| base:Book_Title                                                                             | base:Book              |
| base:Book_Category                                                                          | base:Book              |
| base:Enterance_Exam                                                                         | base:Book_Category     |
| base:Award_Year                                                                             | base:Award             |
| base:Journals                                                                               | base:Research_Category |
| base:Chemistry                                                                              | base:Engineering       |
| base:ICSE                                                                                   | base:school            |
| base:Maths                                                                                  | base:Engineering       |
| base:Security                                                                               | base:Computer_Internet |
| base:English                                                                                | base:Academics         |
| base:NIOS                                                                                   | base:school            |
| base:Physics                                                                                | base:Engineering       |
```

Figure 4.19 Displaying SubClassOf relation

a) If user wishes to know all concepts related to domain Input: *" select distinct ?Concept where {[] a ?Concept}"*
Output: the snapshot in figure 4.20.



ontology - NetBeans IDE 6.9.1
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
<default config>
Output - ontology (run)

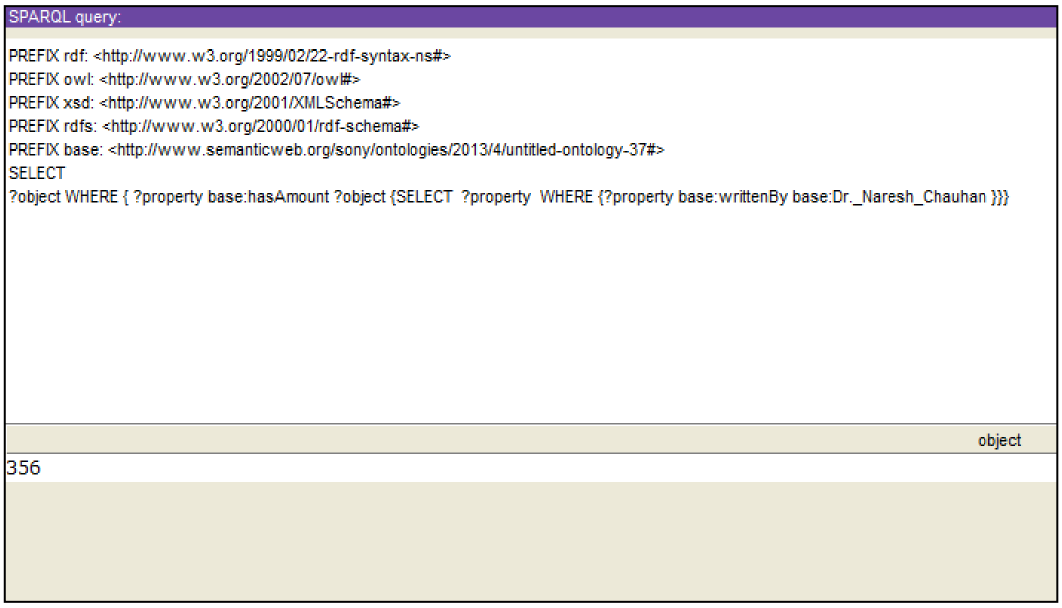
```
run:
<ModelCom   {:amount @rdfs:range xsd:integer; :amount @rdfs:domain :Book_Price; :amount @ace_lexic
"; :0446494712 @ace_lexicon:PN_sg "0446494712"; :0446494712 @rdf:type :Book_ISBN; :0446494712 @rdf:
<ModelCom   {:amount @rdfs:range xsd:integer; :amount @rdfs:domain :Book_Price; :amount @ace_lexic
"; :0446494712 @ace_lexicon:PN_sg "0446494712"; :0446494712 @rdf:type :Book_ISBN; :0446494712 @rdf:
-----------------------------------
| Concept                         |
===================================
| base:Author_Name                |
| owl:Restriction                 |
| owl:IrreflexiveProperty         |
| owl:NamedIndividual             |
| owl:Class                       |
| base:Research_Title             |
| owl:AnnotationProperty          |
| base:Book_Price                 |
| base:International_Conference   |
| owl:FunctionalProperty          |
| base:Book_Title                 |
| base:Published_Year             |
| owl:SymmetricProperty           |
| owl:Ontology                    |
| owl:ObjectProperty              |
| base:Book_ISBN                  |
| owl:DatatypeProperty            |
| owl:TransitiveProperty          |
| base:Software_Testing           |
| base:International_Journal      |
| base:Publication_Name           |
```

Figure 4.20 Concepts of Book Domain OWL

b) If query type is transitive query List the amount of books written by Dr Naresh Chauhan shown in figure 4.21.

Input : *"SELECT ?object WHERE { ?property base:hasAmount ?object {SELECT ?property WHERE {?property base:writtenBy base:Dr._Naresh_Chauhan }}}"*

Output:



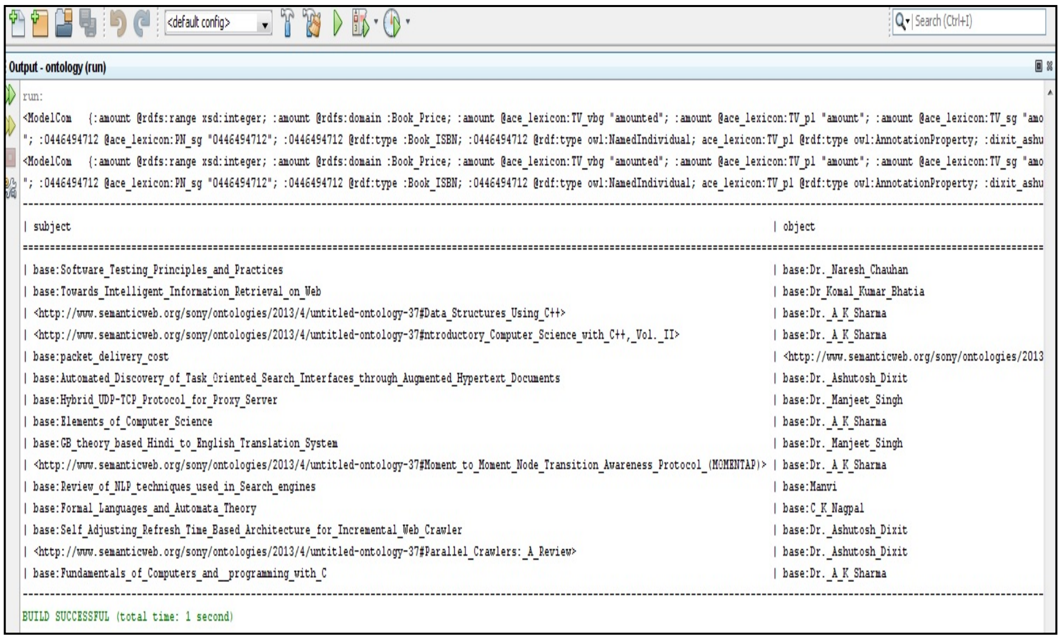
```
SPARQL query:

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX base: <http://www.semanticweb.org/sony/ontologies/2013/4/untitled-ontology-37#>
SELECT
?object WHERE { ?property base:hasAmount ?object {SELECT  ?property  WHERE {?property base:writtenBy base:Dr._Naresh_Chauhan }}}

object
356
```

Figure 4.21 Transitive Query Over Book Domain OWL

c) List the all subjects and objects that are related by writtenBy object property.

Input : *"SELECT ?subject ?object WHERE { ?subject base:writtenBy ?object }"*

Output: in figure 4.22



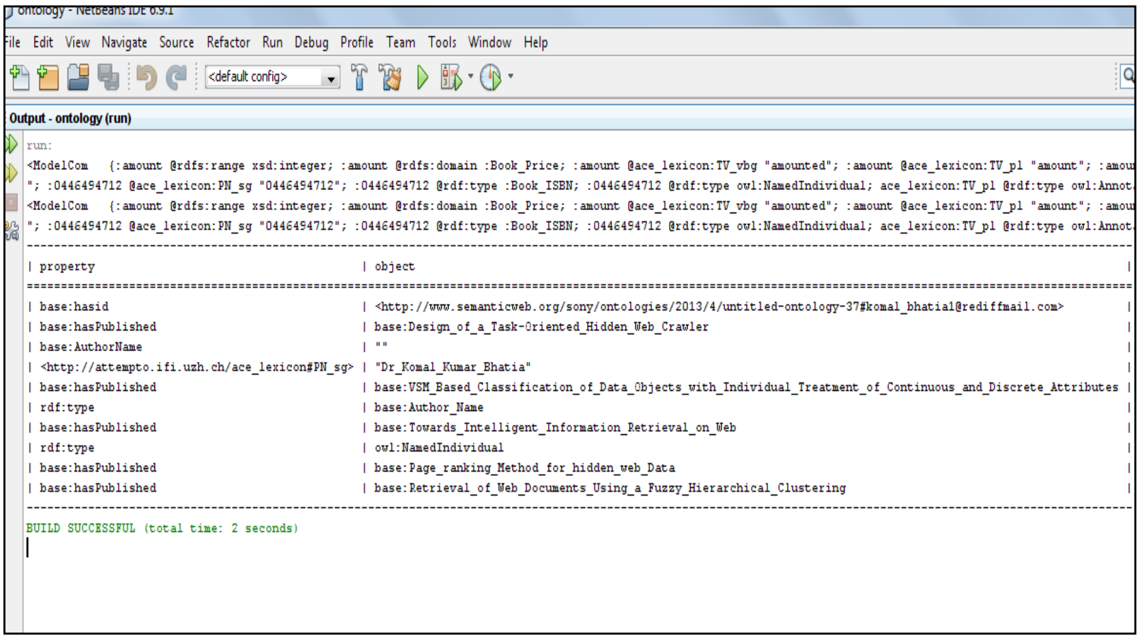
```
Output - ontology (run)

run:
<ModelCom   {:amount @rdfs:range xsd:integer; :amount @rdfs:domain :Book_Price; :amount @ace_lexicon:TV_vbg "amounted"; :amount @ace_lexicon:TV_pl "amount"; :amount @ace_lexicon:TV_sg "amo
"; :0446494712 @ace_lexicon:PN_sg "0446494712"; :0446494712 @rdf:type :Book_ISBN; :0446494712 @rdf:type owl:NamedIndividual; ace_lexicon:TV_pl @rdf:type owl:AnnotationProperty; :dixit_ashu
<ModelCom   {:amount @rdfs:range xsd:integer; :amount @rdfs:domain :Book_Price; :amount @ace_lexicon:TV_vbg "amounted"; :amount @ace_lexicon:TV_pl "amount"; :amount @ace_lexicon:TV_sg "amo
"; :0446494712 @ace_lexicon:PN_sg "0446494712"; :0446494712 @rdf:type :Book_ISBN; :0446494712 @rdf:type owl:NamedIndividual; ace_lexicon:TV_pl @rdf:type owl:AnnotationProperty; :dixit_ashu
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
| subject                                                                                                                                  | object
==========================================================================================================================================================================================
| base:Software_Testing_Principles_and_Practices                                                                                           | base:Dr._Naresh_Chauhan
| base:Towards_Intelligent_Information_Retrieval_on_Web                                                                                    | base:Dr_Komal_Kumar_Bhatia
| <http://www.semanticweb.org/sony/ontologies/2013/4/untitled-ontology-37#Data_Structures_Using_C++>                                       | base:Dr._A_K_Sharma
| <http://www.semanticweb.org/sony/ontologies/2013/4/untitled-ontology-37#ntroductory_Computer_Science_with_C++,_Vol._II>                  | base:Dr._A_K_Sharma
| base:packet_delivery_cost                                                                                                                | <http://www.semanticweb.org/sony/ontologies/2013
| base:Automated_Discovery_of_Task_Oriented_Search_Interfaces_through_Augmented_Hypertext_Documents                                        | base:Dr._Ashutosh_Dixit
| base:Hybrid_UDP-TCP_Protocol_for_Proxy_Server                                                                                            | base:Dr._Manjeet_Singh
| base:Elements_of_Computer_Science                                                                                                        | base:Dr._A_K_Sharma
| base:GB_theory_based_Hindi_to_English_Translation_System                                                                                 | base:Dr._Manjeet_Singh
| <http://www.semanticweb.org/sony/ontologies/2013/4/untitled-ontology-37#Moment_to_Moment_Node_Transition_Awareness_Protocol_(MOMENTAP)> | base:Dr._A_K_Sharma
| base:Review_of_NLP_techniques_used_in_Search_engines                                                                                     | base:Manvi
| base:Formal_Languages_and_Automata_Theory                                                                                                | base:C_K_Nagpal
| base:Self_Adjusting_Refresh_Time_Based_Architecture_for_Incremental_Web_Crawler                                                          | base:Dr._Ashutosh_Dixit
| <http://www.semanticweb.org/sony/ontologies/2013/4/untitled-ontology-37#Parallel_Crawlers:_A_Review>                                     | base:Dr._Ashutosh_Dixit
| base:Fundamentals_of_Computers_and__programming_with_C                                                                                   | base:Dr._A_K_Sharma
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
BUILD SUCCESSFUL (total time: 1 second)
```

Figure 4.22 Writtenby Relation of Book Domain OWL

d) List the properties and object related to Dr. Komal Kumar Bhatia .

Input: *" SELECT ?property ?object WHERE { base:Dr_Komal_Kumar_Bhatia ?property ?object}"*

Output: Figure 4.23


ontology - NetBeans IDE 6.9.1
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
<default config>
Output - ontology (run)

```
run:
<ModelCom   {:amount @rdfs:range xsd:integer; :amount @rdfs:domain :Book_Price; :amount @ace_lexicon:TV_vbg "amounted"; :amount @ace_lexicon:TV_pl "amount"; :amou
"; :0446494712 @ace_lexicon:PN_sg "0446494712"; :0446494712 @rdf:type :Book_ISBN; :0446494712 @rdf:type owl:NamedIndividual; ace_lexicon:TV_pl @rdf:type owl:Annot
<ModelCom   {:amount @rdfs:range xsd:integer; :amount @rdfs:domain :Book_Price; :amount @ace_lexicon:TV_vbg "amounted"; :amount @ace_lexicon:TV_pl "amount"; :amou
"; :0446494712 @ace_lexicon:PN_sg "0446494712"; :0446494712 @rdf:type :Book_ISBN; :0446494712 @rdf:type owl:NamedIndividual; ace_lexicon:TV_pl @rdf:type owl:Annot
------------------------------------------------------------------------------------------------------------------------------------------------------------------
| property                                   | object                                                                                                       |
==================================================================================================================================================================
| base:hasid                                 | <http://www.semanticweb.org/sony/ontologies/2013/4/untitled-ontology-37#komal_bhatia1@rediffmail.com>         |
| base:hasPublished                          | base:Design_of_a_Task-Oriented_Hidden_Web_Crawler                                                            |
| base:AuthorName                            | ""                                                                                                           |
| <http://attempto.ifi.uzh.ch/ace_lexicon#PN_sg> | "Dr_Komal_Kumar_Bhatia"                                                                                  |
| base:hasPublished                          | base:VSM_Based_Classification_of_Data_Objects_with_Individual_Treatment_of_Continuous_and_Discrete_Attributes |
| rdf:type                                   | base:Author_Name                                                                                             |
| base:hasPublished                          | base:Towards_Intelligent_Information_Retrieval_on_Web                                                        |
| rdf:type                                   | owl:NamedIndividual                                                                                          |
| base:hasPublished                          | base:Page_ranking_Method_for_hidden_web_Data                                                                 |
| base:hasPublished                          | base:Retrieval_of_Web_Documents_Using_a_Fuzzy_Hierarchical_Clustering                                        |
------------------------------------------------------------------------------------------------------------------------------------------------------------------
BUILD SUCCESSFUL (total time: 2 seconds)
```

Figure 4.23 Concepts Related to Dr Komal Kumar Bhatia

Many systems have been developed to organize domain specific information into relational table and perform user query over it but with the involvement of Ontology we can create semantic knowledge base of domain specific information. The contribution mainly focuses on process of building Book Domain Ontology for readers and to show how effective the results of queries can be obtains by storing knowledge base in RDF triplets in terms of Ontology.

The ontology created here is used by hidden web crawler to extract the information hidden behind form interfaces by filling the form values from the ontology, discussed in chapter 6.

Chapter 5

GENERATION OF AUTOMATIC ONTOLOGY FROM HIDDEN WEB INTERFACES

5.1 GENERAL

Ontology is a data model defining the set of concepts within a domain and representing an area of knowledge and relationship between those concepts. The construction of ontology for book domain is shown in chapter 4. The various instance values taken to fill the knowledge base have been taken from the examples of faculty from YMCA University, Whereas the hidden web present on WWW is actually very vast and cannot be fully retrieved using these limited values. In this chapter a novel technique for creation of ontology with the help of form pages is proposed and implemented. This technique is novel in the sense that it is using the information present on the search interface to create the ontology. The ontology thus created is stored as the semantic database in Oracle 10g in the form of triples <Subject, Predicate, Object >.

5.2 THE PROPOSED ARCHITECTURE OF AUTOMATIC GENERATION OF ONTOLOGY FROM SEARCH INTERFACES

The Architecture of the proposed system along with different components is shown in figure 5.1 below.

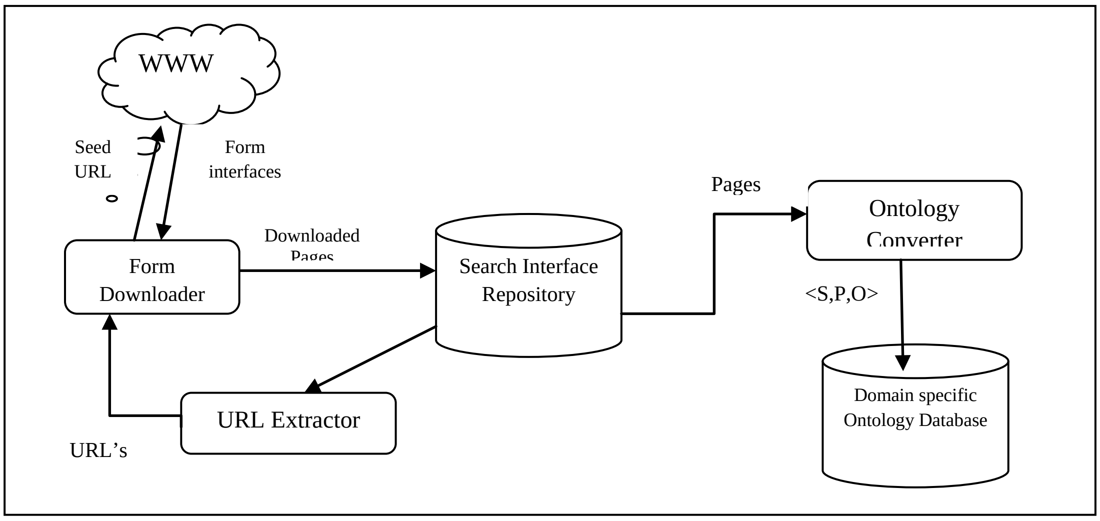


Figure 5.1 The Proposed Architecture of Automatic Generation of Ontology From Search Interfaces.

The architecture consists of following major components:

1. Form Downloader.
2. Search Interface Repository.
3. Ontology Converter.
4. Ontological Database.

The detailed description of each component is given below.

5.2.1 Form downloader: A form downloader is designed which starts with a seed URL and downloads the form pages. Those web pages which are having entry point to hidden web documents i.e. the search interfaces are downloaded and stored. For this the downloader checks from the source code of the web page, whether the page contain the *<form>* tag element or not. If source code of page contains the *<form>* tag i.e. this page is actually the entry point for hidden web is downloaded and taken to the next step. Hence those pages that contain fields to be filled by the user are considered only rest are discarded. The source code of each form page is then stored in the repository which is passed to next component for further processing.

5.2.2 Search Interface repository: This repository contains the downloaded form pages above. These pages are then taken by ontology converter for creating ontology with the help of information present within the tags and between a pair of tags in source code of the page.

5.2.3 Ontology Converter: This component is the core component of the system. It takes the source code of form pages stored in the repository as input and send them to classifier which classifies them according to their format. The classifier further sends them to corresponding analyzers. Each analyzer extracts the meaningful information from various tags present in the page. Using this information ontology is constructed. Different form pages are having different formats like RDF, XML, HTML, OWL etc. According to the format of these pages, they are sent to different analyzing modules in order to create a generic and well populated ontology. Following three type of analyzer modules are proposed:

i) XML/HTML Analyser
ii) RDF Analyser

iii) OWL Analyser

The detailed description of each of these is given in next section. These analyzers extract the required information from each page and convert it in the form of Subject, Object and Predicate triples. These <S,P,O> triples are then stored in domain specific ontology database (DSODB) . The architecture of ontology converter module is shown in figure 5.2.

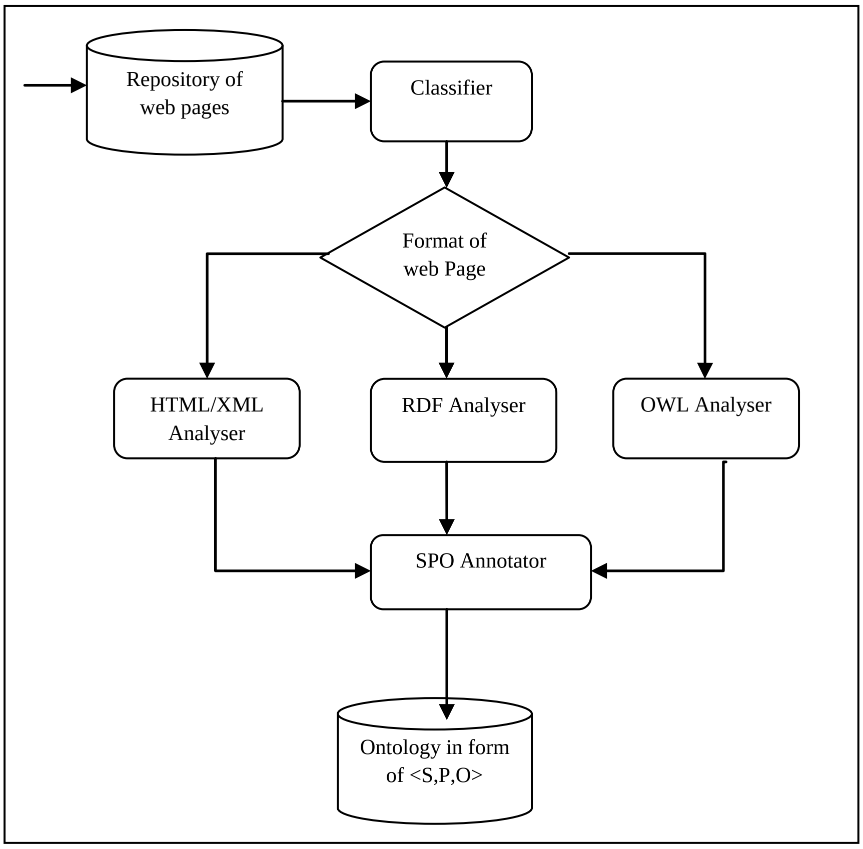


Figure 5.2 The Architecture Showing Components of Ontology Converter Module.

The description of each of its sub components is given below:

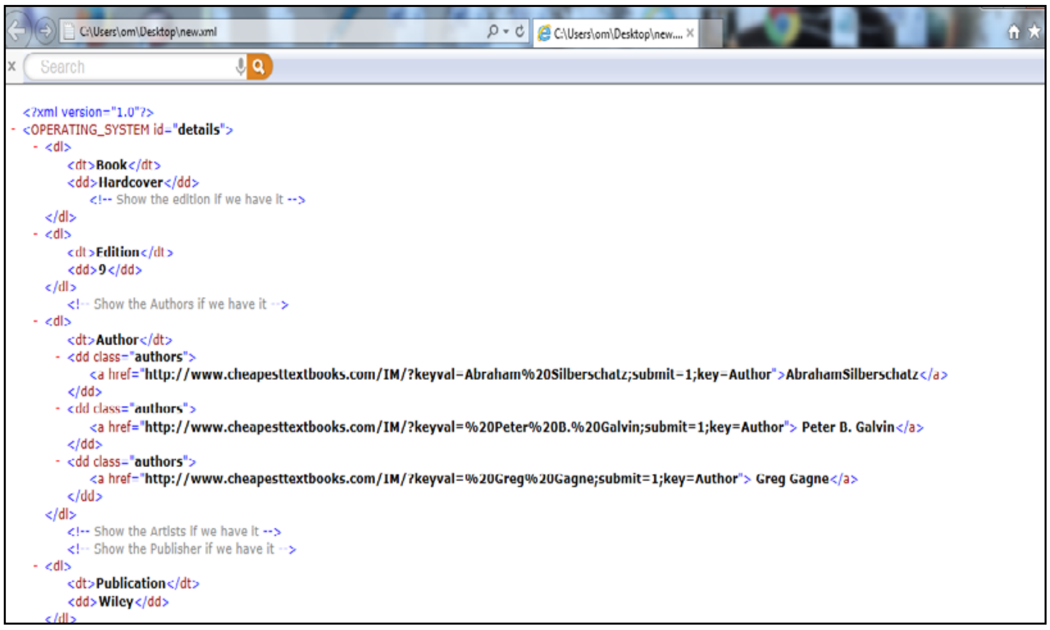
i) Classifier: This component after getting the form pages from repository reads different tags present in source code of the page and classifies them according to their types/format. The different classifications available online till now are: i) HTML ii) XML/HTML iii) RDF/XML iv) OWL/RDFS.

The web pages traditionally are created in HTML format. For dynamic page generation the programmers started using scripts. To solve the serialisation problem XML came. Then for adding more semantics to the text RDF was developed. Above all of them comes is OWL adding more information. The classifier according to the type of web page sends the page to the respective analyser for extracting further information. If page contains Owl tags then is sent to OWL analyser similarly if page contains RDF tag then sends to RDF analyser otherwise will send to XML analyser.

ii) XML Analyser

To analyse XML document DOM parser which is already available is used here. DOM parser parses the entire XML document and loads it into memory, then models it in form of a" Tree structure" for easy traversal or manipulation. After traversing the tree node by node required information is gathered. According to DOM, everything in XML document is in form of nodes. It says that the: i) Entire document is -a document node, ii) Every Xml element is an element node and iii) Every attribute is an attribute node.

For Book domain, here Book is the super class and is stored as Subject. The empty nodes created by <dl> tag is defined as subclasses and also stored as Subject. Label to empty nodes like author, title, price, ISBN, publication etc is given by <dt > tag. Instances/values of these subclasses is found by<dd> tag and are treated as Object. This process of extracting <dt> and dd tag continues until the count of dl tag does not get null value and nlist variable is empty.



```
  <?xml version="1.0"?>
- <OPERATING_SYSTEM id="details">
  - <dl>
      <dt>Book</dt>
      <dd>Hardcover</dd>
        <!-- Show the edition if we have it -->
    </dl>
  - <dl>
      <dt>Edition</dt>
      <dd>9</dd>
    </dl>
      <!-- Show the Authors if we have it -->
  - <dl>
      <dt>Author</dt>
    - <dd class="authors">
        <a href="http://www.cheapesttextbooks.com/IM/?keyval=Abraham%20Silberschatz;submit=1;key=Author">AbrahamSilberschatz</a>
      </dd>
    - <dd class="authors">
        <a href="http://www.cheapesttextbooks.com/IM/?keyval=%20Peter%20B.%20Galvin;submit=1;key=Author"> Peter B. Galvin</a>
      </dd>
    - <dd class="authors">
        <a href="http://www.cheapesttextbooks.com/IM/?keyval=%20Greg%20Gagne;submit=1;key=Author"> Greg Gagne</a>
      </dd>
    </dl>
      <!-- Show the Artists if we have it -->
      <!-- Show the Publisher if we have it -->
  - <dl>
      <dt>Publication</dt>
      <dd>Wiley</dd>
    </dl>
```

Figure 5.3 Example Source Code from the Website www.Cheapesttestbooks.Com

Figure 5.4 shows the example of XML tags and conversion in Graph for source code of Book domain from the website named *www.cheapesttestbooks.com* (figure 5.3) and used for extraction of data from various tags and meta tags defined in XML/HTML file in.

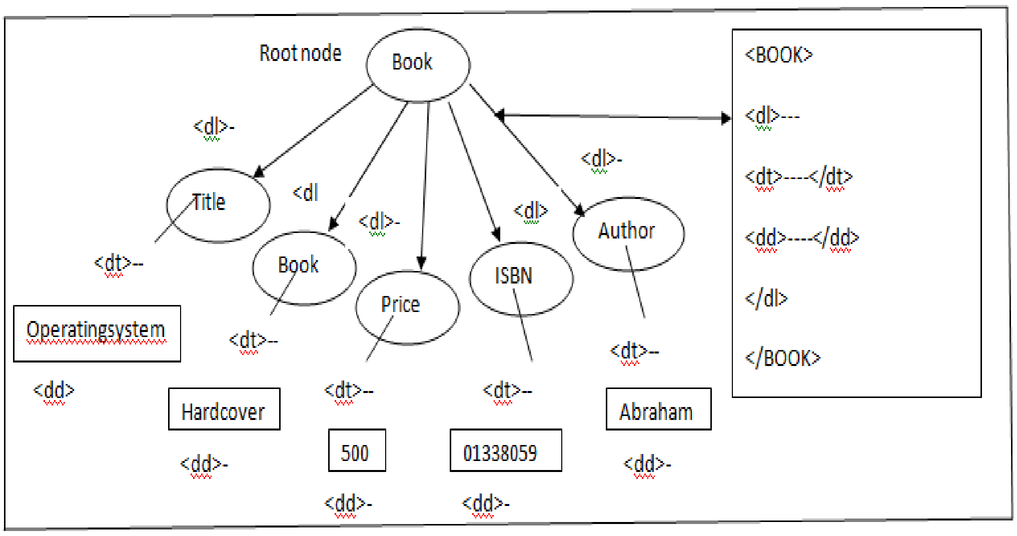


Figure 5.4 Tree Structure by DOM Parser for XML Code Shown in Square Box.

Here for XML documents according to algorithm firstly the root node is constructed and then empty nodes are constructed by *<dl>* tags. Then *<dt><dd>* tags are taken and information is extracted and saved in database

- **Procedure for finding <S,P,O> triple from XML page format using DOM parser is as follows:**

  ➢ **Finding Subject**

  i) The method *getnodname()* is used to find Subject. The value returned by this function is stored as subject. This is the root element.

  ii) The method *getelementbytagname(dl)* for <dl > tags is used to extract the number of children nodes. This provides a number of empty nodes. All these nodes are Subject.

  iii) The labels of the nodes found above are extracted using *getelementsbytagname(dt*) method for <dt> tag. All these values are stored as Subject.

- **Finding Object**

  Use method *getelementsbytagname(dd)* for finding objects. The values returned by this function are treated as objects for above extracted subject values.

- **Finding Predicate**

  The method *getattribute()* is used to extract information associated with labels and values. This information is used to find the relationship between labels and values and is stored as predicate.

The flow chart showing the working of XML analyser is summarised in figure 5.5 below.

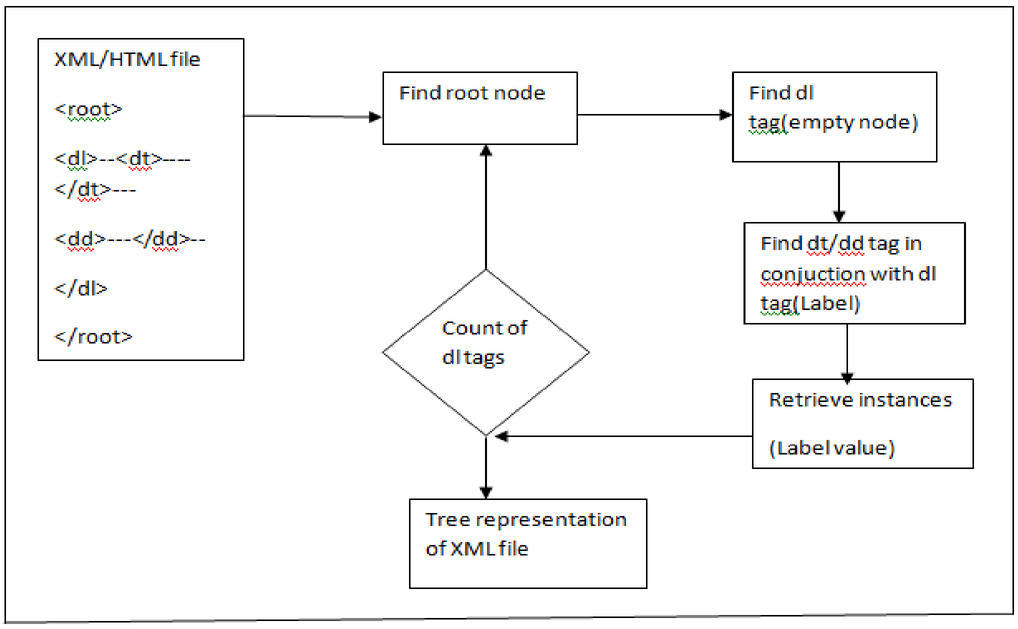
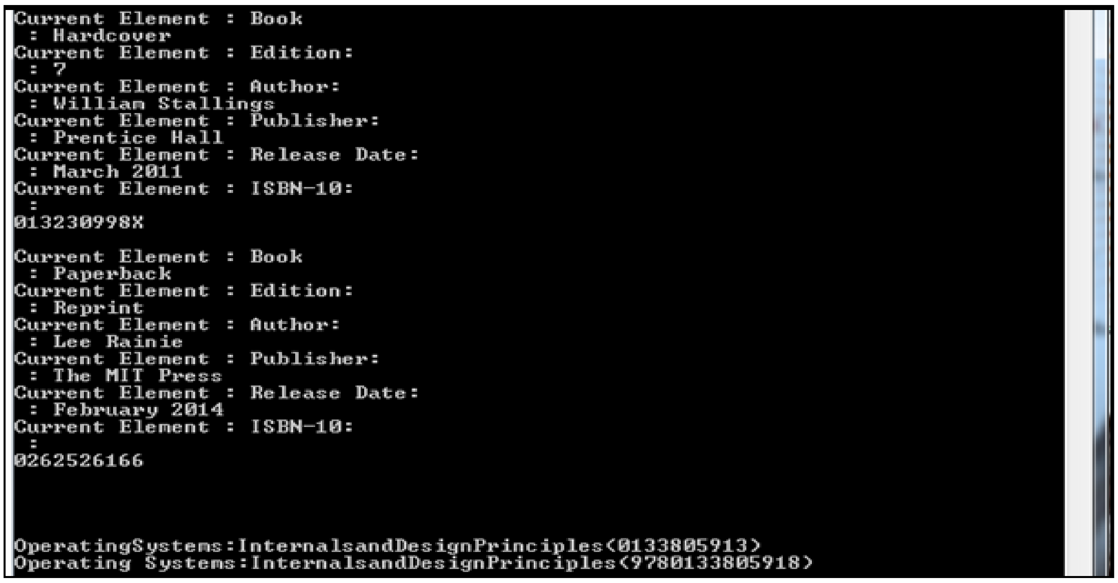


Figure 5.5 Flow Chart Showing the Working of XML Analyser.

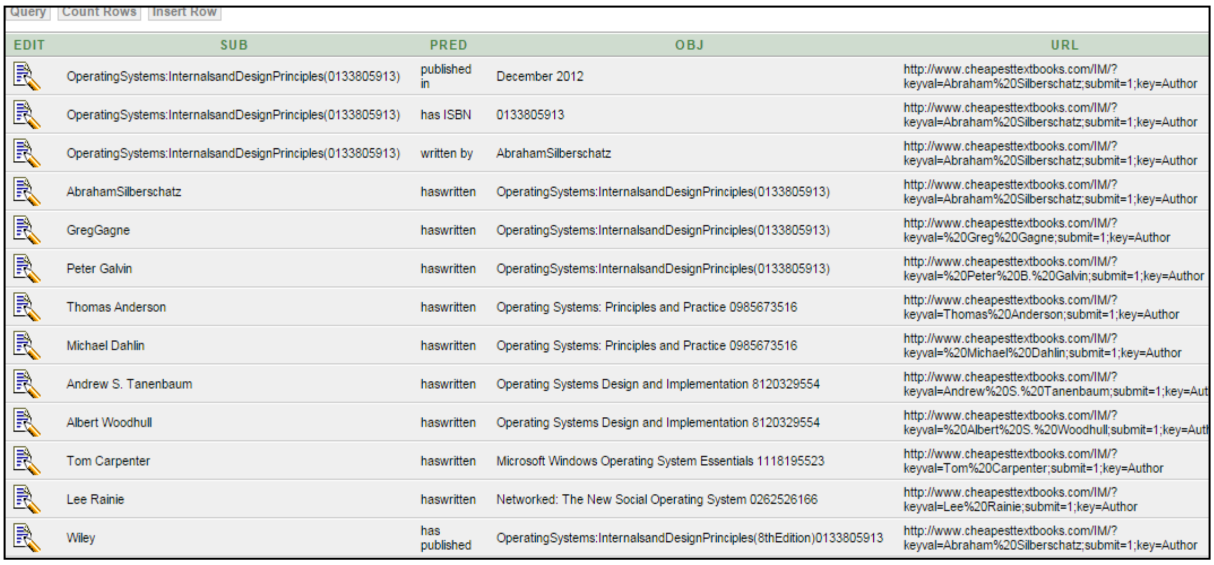
- **Storing Extracted Values in Semantic Database**: Extraction of <SPO> triples is done where xml file is taken as input file and traversed as nodes of tree step by step as shown in figure 5.6. First of all, root node is found which is <BOOK> here and treated as super class. Then number of children nodes has been extracted and created like Author, Title, Price, ISBN, Publication etc. The subclasses and further, instances of various subclasses have been created. All the classes, subclasses and instances are then finally stored in database in form of <S,P,O>.



```
Current Element : Book
 : Hardcover
Current Element : Edition:
 : 7
Current Element : Author:
 : William Stallings
Current Element : Publisher:
 : Prentice Hall
Current Element : Release Date:
 : March 2011
Current Element : ISBN-10:
 :
013230998X

Current Element : Book
 : Paperback
Current Element : Edition:
 : Reprint
Current Element : Author:
 : Lee Rainie
Current Element : Publisher:
 : The MIT Press
Current Element : Release Date:
 : February 2014
Current Element : ISBN-10:
 :
0262526166



OperatingSystems:InternalsandDesignPrinciples(0133805913)
Operating Systems:InternalsandDesignPrinciples(9780133805918)
```

Figure 5.6 Snapshot of extraction of <S,P,O> triples from XML/HTML File

In figure 5.7 below <subject,predicate,object> with their corresponding urls find in source file has been saved in database. The values associated as Subject and Object depends upon the predicate stored with them. For one predicate one value may behave as subject and for other predicate the same value may behave as object. For example"Abraham haswritten operatingsystems and internalsdesignPrinciples", Here Abraham is subject and operatingsystems and internalsdesignPrinciples are objects. But for predicate writtenby, operatingsystems and internalsdesignPrinciples became subject and Abraham became object.



EDIT	SUB	PRED	OBJ	URL
✎	OperatingSystems:InternalsandDesignPrinciples(0133805913)	published in	December 2012	http://www.cheapesttextbooks.com/IM/?keyval=Abraham%20Silberschatz;submit=1;key=Author
✎	OperatingSystems:InternalsandDesignPrinciples(0133805913)	has ISBN	0133805913	http://www.cheapesttextbooks.com/IM/?keyval=Abraham%20Silberschatz;submit=1;key=Author
✎	OperatingSystems:InternalsandDesignPrinciples(0133805913)	written by	AbrahamSilberschatz	http://www.cheapesttextbooks.com/IM/?keyval=Abraham%20Silberschatz;submit=1;key=Author
✎	AbrahamSilberschatz	haswritten	OperatingSystems:InternalsandDesignPrinciples(0133805913)	http://www.cheapesttextbooks.com/IM/?keyval=Abraham%20Silberschatz;submit=1;key=Author
✎	GregGagne	haswritten	OperatingSystems:InternalsandDesignPrinciples(0133805913)	http://www.cheapesttextbooks.com/IM/?keyval=%20Greg%20Gagne;submit=1;key=Author
✎	Peter Galvin	haswritten	OperatingSystems:InternalsandDesignPrinciples(0133805913)	http://www.cheapesttextbooks.com/IM/?keyval=%20Peter%20B.%20Galvin;submit=1;key=Author
✎	Thomas Anderson	haswritten	Operating Systems: Principles and Practice 0985673516	http://www.cheapesttextbooks.com/IM/?keyval=Thomas%20Anderson;submit=1;key=Author
✎	Michael Dahlin	haswritten	Operating Systems: Principles and Practice 0985673516	http://www.cheapesttextbooks.com/IM/?keyval=%20Michael%20Dahlin;submit=1;key=Author
✎	Andrew S. Tanenbaum	haswritten	Operating Systems Design and Implementation 8120329554	http://www.cheapesttextbooks.com/IM/?keyval=Andrew%20S.%20Tanenbaum;submit=1;key=Aut
✎	Albert Woodhull	haswritten	Operating Systems Design and Implementation 8120329554	http://www.cheapesttextbooks.com/IM/?keyval=%20Albert%20S.%20Woodhull;submit=1;key=Aut
✎	Tom Carpenter	haswritten	Microsoft Windows Operating System Essentials 1118195523	http://www.cheapesttextbooks.com/IM/?keyval=Tom%20Carpenter;submit=1;key=Author
✎	Lee Rainie	haswritten	Networked: The New Social Operating System 0262526166	http://www.cheapesttextbooks.com/IM/?keyval=Lee%20Rainie;submit=1;key=Author
✎	Wiley	has published	OperatingSystems:InternalsandDesignPrinciples(8thEdition)0133805913	http://www.cheapesttextbooks.com/IM/?keyval=Abraham%20Silberschatz;submit=1;key=Author

Figure 5.7 Snapshot of Saved <S,P,O> Triples in Oracle Database for XML/HTML File

iii) RDF Analyser: RDF is a standard for describing resources. It is similar to conceptual models such as entity-relationship model. It is based upon idea of making statements about resources in form of subject-predicate-object expressions. These expressions are known as triples in terms of RDF. The subject denotes the resource and predicate denotes relationships of resource between subject and object. For example “Book on operating system” can be represented in RDF as triple where ‘book’ denoting subject, ‘on’ denoting predicate, ’operating system’ denoting object**.**

The subject of an RDF statement is either a URI or blank node, both of which denote resources. Resources indicated by blank nodes are called anonymous resources. The predicate is a URI which also represents a resource relationship. The object can also be a URI, blank node or string literal. For example given below are the two ways to specify RDF. Where Figure 5.8 a) represent RDF/XML format and figure 5.8 b) represents the pure RDF format.

```
<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-
rdf-syntax-ns#"
xmlns:dc="http://purl.org/dc/elements/1.1/">
<rdf:Description
rdf:about="http://www.w3.org/">
<dc:title>World          Wide          Web
Consortium</dc:title>   </rdf:Description>
</rdf:RDF>
```

```
<rdf:Property rdf:about="isbn" rdfs:la
bel="ISBN
Number" rdfs:comment="The ISBN
number of the book.">
</rdf:Property>
<rdf:Property  rdf:about="authoredBy"
  rdfs:label="Authored
By"  rdfs:comment="An author of this
book.">
```

Figure 5.8 a) RDF/XML Format **b) Pure RDF Format**

Where URI is string of characters used to identify a name of resource, such identification enables interaction with representations of resource over network (WWW).The most common form of URI is web page address. Basically, URI describes:

1) The mechanism used to access resources.
2) Computer that contains resource.
3) A file name of resource on computer.
4) vCard ontology focus on describing people and organizations including location info & group of such entities .vCard ontology use http://www.w3.org/2006/vcard/ns# namespace URI for representing vCard objects in RDF.

RDF analyser proposed here, after getting the attached RDF of the web page analyses its content and converts the data into <SPO> triple according to the procedure defined below. RDF is usually embedded in an <rdf:RDF> tag element. Then there is an <rdf:Description> tag which describes the resource whose URI is defined in <rdf:about> tag and is represented as root node and becomes SUBJECT of <SPO>. If the <rdf:about> is missing, the element would represent a blank node. Further, the tags enclosed within description tag may have different names depending upon the page like <rdf:property>, <dc>, <vCard:FN> which describes a property of resource whose URI is described in rdf:about[ ] .

The property is treated as the OBJECT of <SPO> e.g. in above figure 5.8 a) the description tag is associated with *www.w3.org* having title as World wide web Consortium hence www.w3.org is treated as Subject and title is treated as Object. The dc or property tag may contain a label and/or a comment associated with the corresponding attribute e.g. title and label tags in above figures respectively.

In Figure 5.8 b) in the property tag we have label as ISBN treated as object of rdf:about this rdf property is associated with some rdf :class which is book. Hence book is subject and isbn is object here.

- **Procedure for finding <SPO> in RDF format**

  - **Finding Subject**
    Find <rdf:Description> tag element in <rdf:RDF> tag. Extract <rdf:about> tag in <rdf:Description> tag to define resource of information and assigns it as root node, The first element of Subject Set S.

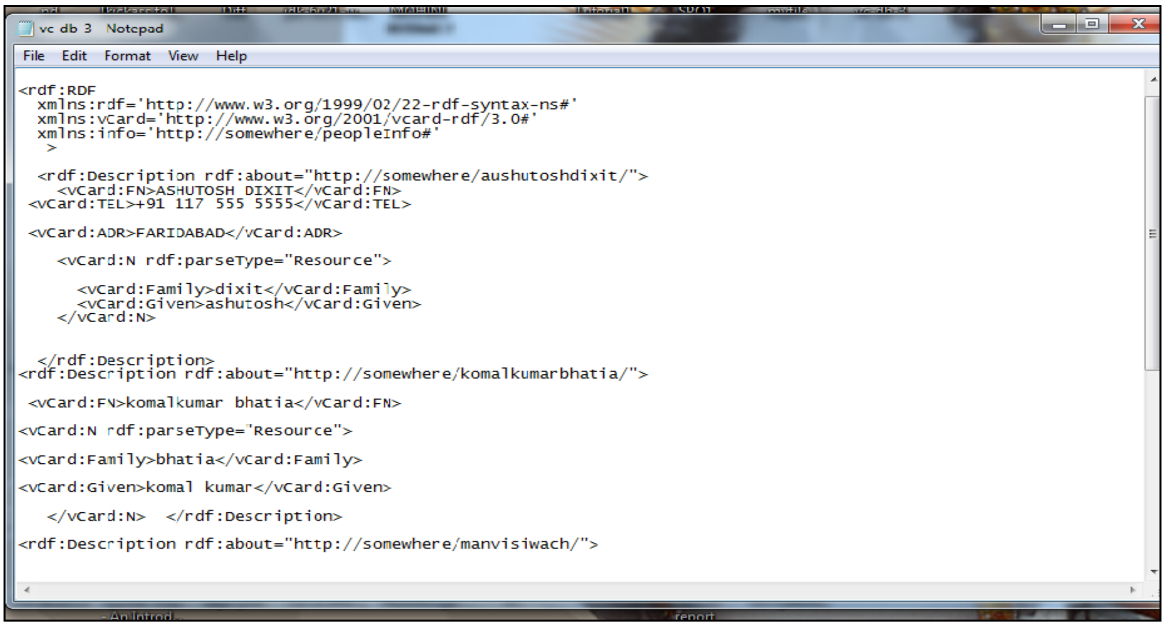
  - **Finding Object**
    Extract </dc> or </rdf:property>or <vCard:FN> information enclosed in between these tags and treat them as object value for defined property. Here dc tag contain the value after : which is treated as object and property tag contain other tags like comment and label, label is treated as object and comment can be used to find the predicate. The values find are added to set O and P correspondingly.

- **Finding Predicate**

  Find <dc>, <rdf:property>,<vCard:FN> tags to describe property of resource and store them as Predicate where <dc:> and <rdf: property> specifies instance or attribute property, FN is property Name in vCard namespace. These tags contain information that will help to find the relationship between S and O.

As it is difficult to find out relation/predicate from RDF document the <*dc*/property> tag value is useful for the same i.e. to find out the Predicate of <SPO>. In case of <dc> tag it contains two things: 1. The attribute defined after colon ":" 2. text between <dc> and </dc>.

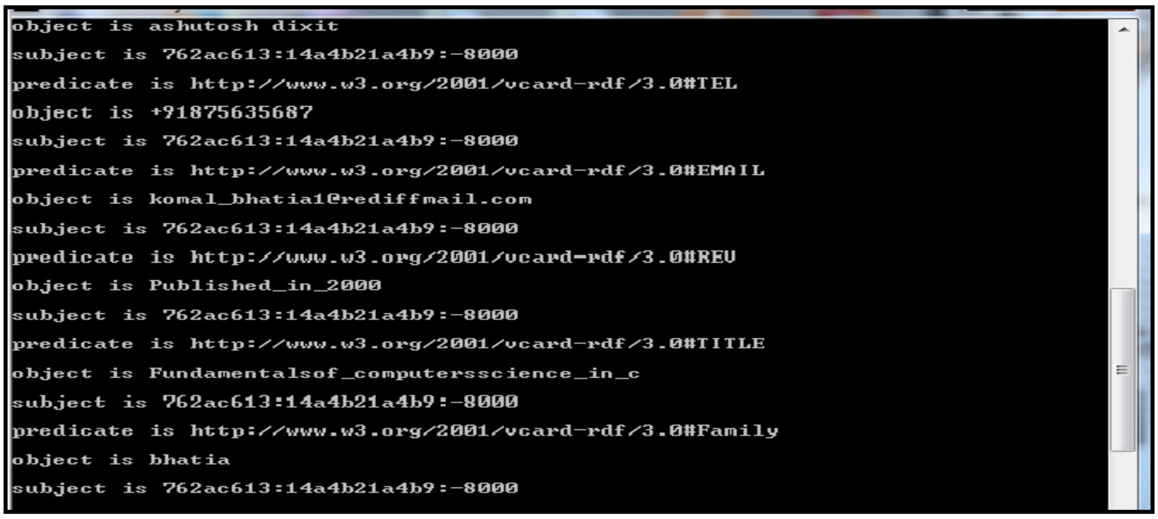
After extracting these two things from <dc> the relation between element in <rdf:description> and attribute in <dc> is found. In case of <rdf:property> this tag may contain a label and comment tag which is parsed and used to find out the predicate. Again there are not many live web pages defined in RDF format till now as research is still going on, hence in this work the example URL http://www.w3.org/2001/vcard-rdf#FN[] is taken as input (figure 5.9).



```
<rdf:RDF
  xmlns:rdf='http://www.w3.org/1999/02/22-rdf-syntax-ns#'
  xmlns:vCard='http://www.w3.org/2001/vcard-rdf/3.0#'
  xmlns:info='http://somewhere/peopleInfo#'
   >

  <rdf:Description rdf:about="http://somewhere/aushutoshdixit/">
    <vCard:FN>ASHUTOSH DIXIT</vCard:FN>
 <vCard:TEL>+91 117 555 5555</vCard:TEL>

 <vCard:ADR>FARIDABAD</vCard:ADR>

    <vCard:N rdf:parseType="Resource">

      <vCard:Family>dixit</vCard:Family>
      <vCard:Given>ashutosh</vCard:Given>
    </vCard:N>


  </rdf:Description>
<rdf:Description rdf:about="http://somewhere/komalkumarbhatia/">

 <vCard:FN>komalkumar bhatia</vCard:FN>

<vCard:N rdf:parseType="Resource">

<vCard:Family>bhatia</vCard:Family>

<vCard:Given>komal kumar</vCard:Given>

   </vCard:N>  </rdf:Description>

<rdf:Description rdf:about="http://somewhere/manvisiwach/">
```

Figure 5.9 Example Source Code Taken for RDF Format

Extraction of RDF triples is done from input file where <rdf:Description> describes resource or subject and enclosed meta tags within <rdf:Description> tag defines property or predicate and object is literal value between each associated meta tags as shown in figure 5.10.



```
object is ashutosh dixit
subject is 762ac613:14a4b21a4b9:-8000
predicate is http://www.w3.org/2001/vcard-rdf/3.0#TEL
object is +91875635687
subject is 762ac613:14a4b21a4b9:-8000
predicate is http://www.w3.org/2001/vcard-rdf/3.0#EMAIL
object is komal_bhatia1@rediffmail.com
subject is 762ac613:14a4b21a4b9:-8000
predicate is http://www.w3.org/2001/vcard-rdf/3.0#REV
object is Published_in_2000
subject is 762ac613:14a4b21a4b9:-8000
predicate is http://www.w3.org/2001/vcard-rdf/3.0#TITLE
object is Fundamentalsof_computersscience_in_c
subject is 762ac613:14a4b21a4b9:-8000
predicate is http://www.w3.org/2001/vcard-rdf/3.0#Family
object is bhatia
subject is 762ac613:14a4b21a4b9:-8000
```

Figure 5.10 Snapshot of Extraction of RDF Triples

iv) OWL Analyser: OWL format consist of RDFS tags as depicted in figure 5.11. One can easily find the class, subclass and property tags from this. Also there are range and domain tags available in the code itself. Few terms are used interchangeably below but have same meaning/context like: Element=concept=class, Property=attribute=relation, Entity=object=instance.

But there is not much information available online in OWL format currently i.e. there is not any form page available in OWL format online. e.g. The URL http://ebiquity.umbc.edu/ontology/publication.owl taken from SWOOGLE[103] search engine is in OWL format figure 5.11, but it is not a form page.

```
<owl:Ontology rdf:about="http://ebiquity.umbc.edu/ontology/publication.owl#publication"
>
   <owl:versionInfo>0.1</owl:versionInfo>
<owl:Class rdf:about= "http://swrc.ontoware.org /ontology#InBook">
*1 <rdfs:subClassOf>
    <owl:Class rdf:about=”http://swrc.ontoware.org/ontology#Publication”/>
             </rdfs:subClassOf>

*2<rdfs:subClassOf>
   <owl:Restriction>
 <owl:onProperty rdf:resource="http://swrc.ontoware.org/ontology#publisher"/>
             <owl:allValuesFrom>
             <owl:Class rdf:about="http://swrc.ontoware.org/ontology#Organization"/>
             </owl:allValuesFrom>
             </owl:Restriction>
</rdfs:subClassOf>*2
```

Figure 5.11 Example of Web Page in OWL.

As OWL format also contain RDF tags, one can use the same algorithm as for RDF pages to annotate OWL pages.The parser reads the content of the web pages, filters for various tags and give as output the subject, object and predicate of <S,P,O> triples. The procedure for the same is discussed on below.

- **Procedure for finding <S,P,O> triple from OWL page format is as follows:**
  - **Finding Subject**
    1. i) Find the **rdf:about** tag, this contains the root of the page i.e. the main class. This specifies the URL of the page we are going to consider for parsing. This is stored as subject of <S,P,O> for which next predicate and object will be discovered. Also added to a set U.
    2. ii) Find the **rdf:class** tags which define the parent of all children nodes and itself is the child of root node. The required information attached with class tag may be a URI or an ID as specified in figure above. Class specifies the element about which we are talking currently. An OWL page may contain more than one class tags. Hence all becomes the **Subject** in triples. All these attributes extracted as subject are added to a set **S** of subjects.

  - **Finding Object**
    1. i) The **subclassof** tag in between <owl:class> tells about two things in an OWL code. i) If the subclassof tag contains <Owl:class> tag then this specifies that the above said class in <rdf:about> tag is the subclass of resource specified in <Owl:class> tag***1**. It means the **about resource** is having an is-a relation with the O**wl:class**. All the above said < rdf:about> elements are treated as **object** and <Owl:class > are taken as subject with is_a relation. They further are treated as subject in case ii.
    2. ii) The number of subclass tags enclosed in a class tag having property as <owl:restriction> tag defines the number of children and the name of each child node can be extracted from the **onproperty/label** tag. These will be treated as OBJECT in SPO having has relationship with <rdf:about> in Owl:class. The attributes extracted above are stored in a set O of objects.

- **Finding Predicate**

  Two relations have been derived from step 3 is-a.For deriving more relations the data found in various other tags like <owl:dataproperty> <rdfs:**domain/range/label/id etc.>** is parsed and required meaningful information is extracted and used. The relations extracted are also stored in a set P of predicates.

The data extracted from all steps above have been stored in different sets S, P and O. These sets individually are given for further processing to SPO annotator like removing duplicates stemming, lemmatization etc for getting cleaner information. There is not a single Search interface present in OWL format till now. In future if the dynamic web pages are constructed in OWL this algorithm can be directly used to save the data in <SPO> format.

As there is not much information available in the form of web pages for OWL format, the above algorithm is designed so as to cover as many things as possible. In future new tags may come upon hence the algorithm can be extended for the same.

5.2.4 SPO Annotator: This module after getting the attribute values from the classifier processes them and stores them because the values can not be directly used. The attributes values found above are specific to particular web page they should be converted in to more general forms For example one interface may contain author and other may contain authors for the book domain so s from the second is truncated and stored as author itself. Also to avoid repetition only one attribute is saved with the same name.

```
Algorithm SPO _Annotator

Step 1: Get the set S and O of Subject and Object respectively from classifier
     2:  For each $S_i$€ S and $O_i$ €O do
                  i)      Remove special symbols like: (, -, _, @, $, &,#, ?,   !, *,etc.) and
                          make two elements connected by special symbols.
                  ii)     Remove duplicated in S and O.
                  iii)    Expand abbreviations if any.
     3: Extend the synonym values above of both sets by utilizing WordNet and expand
        the sets.
    4: Find Relation/Predicate if not present and store in P.
    5: Send to database for storage.
```

Figure 5.12 Algorithm for SPO Annotator

One thing here to be noted is that simple comparison of sets is employed with syntactic analysis only. No semantic comparison is applied here at this step. Some attributes may contain some special symbols like ‘leaving_from’,some may contain plural form like ‘adults’ and others may contain some abbreviation e.g. dept_date for departure date etc. Keeping these things in mind algorithm to annotate various terms is designed and applied to each element of both sets S and O. E.g. one interface may contain ‘leaving_from’ and other may contain ‘departure from’. Firstly all the values are stored as elements of the set ‘leaving_from and departure, from ’. The special symbols like underscore from leaving_from is removed and leaving and from are disconnected as two elements in step 2(i). As from comes two times hence duplicate term is removed in step 2(ii). In step 2(iii) if an attribute contains some abbreviation then it is expanded. In step (iv), Wordnet is used to

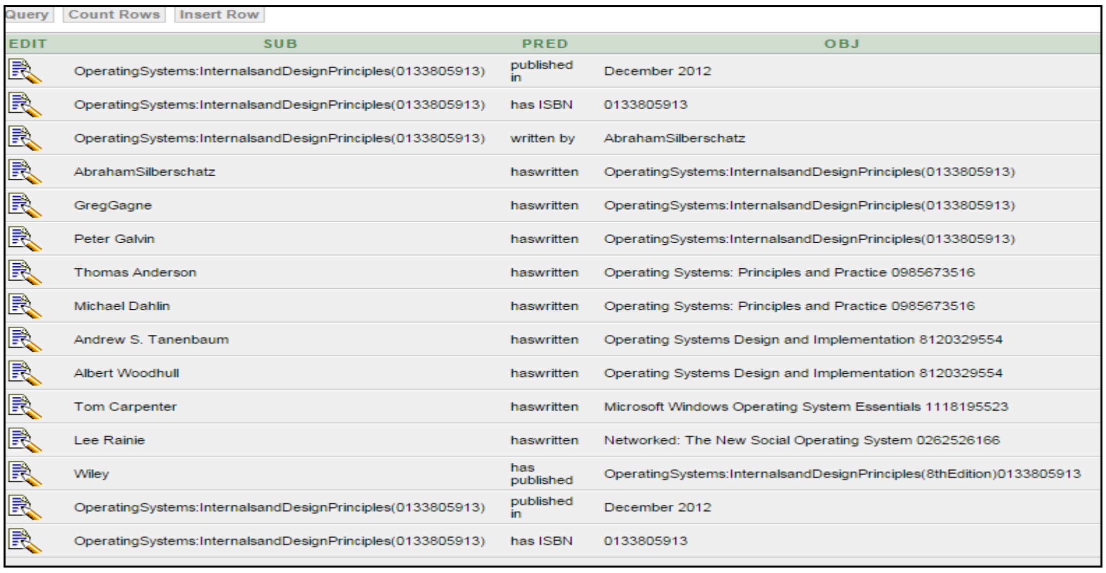
i) Extend the attributes with the help of Synonyms present.
ii) To remove improper words, two rules are used to extend the attributes.

First, for each individual word of a candidate attribute if word has a noun meaning or is a preposition then the word will be kept, otherwise, the word is discarded. E.g. “from” and “to” will be taken as they come under proposition. All state city country name in case of flight domain come under noun and are saved. For book domain writer author and their names are nouns saved.

5.3 Implementation and Evaluation of developed ontology

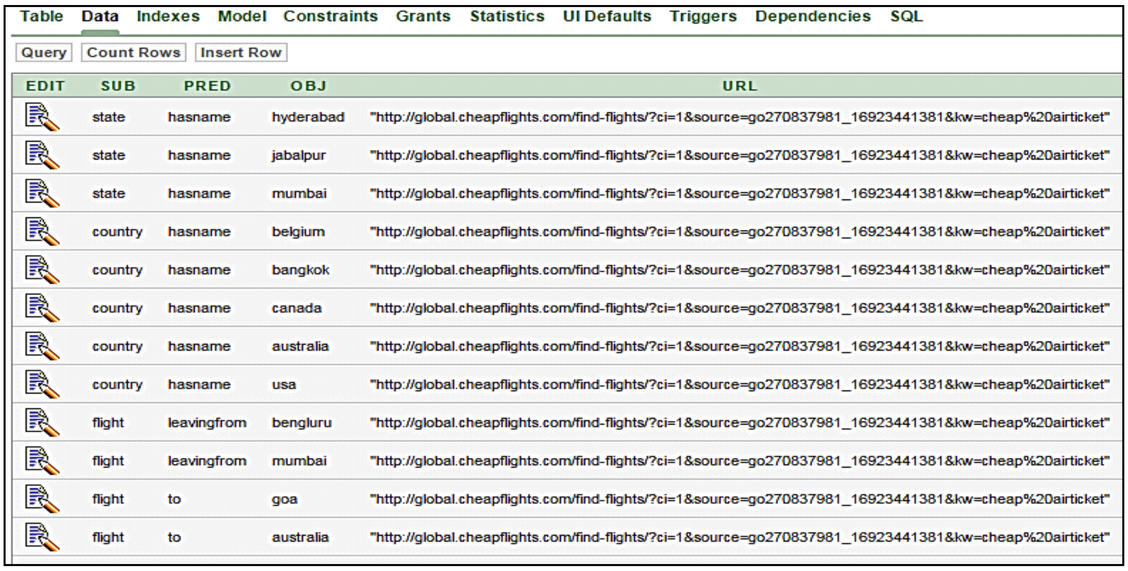
This section gives the snapshots of various instance values of ontology stored in semantic database in oracle in form of <S,P,O> triples. The ontology constructed has also been evaluated on different metrics shown under evaluation sub section.

i) **Implementation:** The values after all this processing are stored in database in form of Subject,Object, Predicate tuple in oracle 10 g using JENA API. JENA is a java API which can be used to create and manipulate RDF graphs. Form pages of two domains i) Book and ii) Airline domain are taken and information retrieved from various interfaces is stored in semantic database Oracle 10g as shown in figure 5.13 and figure 5.14 respectively.



EDIT	SUB	PRED	OBJ
	OperatingSystems:InternalsandDesignPrinciples(0133805913)	published in	December 2012
	OperatingSystems:InternalsandDesignPrinciples(0133805913)	has ISBN	0133805913
	OperatingSystems:InternalsandDesignPrinciples(0133805913)	written by	AbrahamSilberschatz
	AbrahamSilberschatz	haswritten	OperatingSystems:InternalsandDesignPrinciples(0133805913)
	GregGagne	haswritten	OperatingSystems:InternalsandDesignPrinciples(0133805913)
	Peter Galvin	haswritten	OperatingSystems:InternalsandDesignPrinciples(0133805913)
	Thomas Anderson	haswritten	Operating Systems: Principles and Practice 0985673516
	Michael Dahlin	haswritten	Operating Systems: Principles and Practice 0985673516
	Andrew S. Tanenbaum	haswritten	Operating Systems Design and Implementation 8120329554
	Albert Woodhull	haswritten	Operating Systems Design and Implementation 8120329554
	Tom Carpenter	haswritten	Microsoft Windows Operating System Essentials 1118195523
	Lee Rainie	haswritten	Networked: The New Social Operating System 0262526166
	Wiley	has published	OperatingSystems:InternalsandDesignPrinciples(8thEdition)0133805913
	OperatingSystems:InternalsandDesignPrinciples(0133805913)	published in	December 2012
	OperatingSystems:InternalsandDesignPrinciples(0133805913)	has ISBN	0133805913

Figure 5.13 Database in Form of SPO for Book Domain



EDIT	SUB	PRED	OBJ	URL
	state	hasname	hyderabad	"http://global.cheapflights.com/find-flights/?ci=1&source=go270837981_16923441381&kw=cheap%20airticket"
	state	hasname	jabalpur	"http://global.cheapflights.com/find-flights/?ci=1&source=go270837981_16923441381&kw=cheap%20airticket"
	state	hasname	mumbai	"http://global.cheapflights.com/find-flights/?ci=1&source=go270837981_16923441381&kw=cheap%20airticket"
	country	hasname	belgium	"http://global.cheapflights.com/find-flights/?ci=1&source=go270837981_16923441381&kw=cheap%20airticket"
	country	hasname	bangkok	"http://global.cheapflights.com/find-flights/?ci=1&source=go270837981_16923441381&kw=cheap%20airticket"
	country	hasname	canada	"http://global.cheapflights.com/find-flights/?ci=1&source=go270837981_16923441381&kw=cheap%20airticket"
	country	hasname	australia	"http://global.cheapflights.com/find-flights/?ci=1&source=go270837981_16923441381&kw=cheap%20airticket"
	country	hasname	usa	"http://global.cheapflights.com/find-flights/?ci=1&source=go270837981_16923441381&kw=cheap%20airticket"
	flight	leavingfrom	bengluru	"http://global.cheapflights.com/find-flights/?ci=1&source=go270837981_16923441381&kw=cheap%20airticket"
	flight	leavingfrom	mumbai	"http://global.cheapflights.com/find-flights/?ci=1&source=go270837981_16923441381&kw=cheap%20airticket"
	flight	to	goa	"http://global.cheapflights.com/find-flights/?ci=1&source=go270837981_16923441381&kw=cheap%20airticket"
	flight	to	australia	"http://global.cheapflights.com/find-flights/?ci=1&source=go270837981_16923441381&kw=cheap%20airticket"

Figure 5.14 Database in form Of SPO for Airline Domain

iii) **Evaluation:** The developed ontology is evaluated using various metrics of ontology evaluation in table 5.1 below. The metrics used here are already defined in works [104,105].The evaluation shows that the ontology developed is complete, rich and computationaly efficient.

Table 5.1. Evaluation of Proposed Ontology

Evaluation Perspective	Metric	Measure	Statistics of proposed ontology
Ontology Correctness	Pragmatic quality(Accuracy, Relevance)	Relevance is whether the ontology satisfies the specific requirements. Precision: total number correctly found over whole knowledge defined in ontology Recall: total correctly found over all knowledge that should be found	The ontology developed here gives accurate and relevant result as per the query**.**
	Completeness	Coverage	Complete. The ontology developed here is complete in the sense that it tries to cover all possible properties and axioms of book domain.
Ontology Quality	Computational efficiency	Size	The Ontology here makes a complete Binary tree on which the computation cost is lg(n) as compared to the traditional ontology where the computation was of the order O(n).
	Syntactic quality(Richness)	the proportion of features in the ontology language that have been used in an ontology	The richness of ontology developed here is very good. Our ontology includes terms and axioms both hence is richer.
	Semantic quality(Interpretability, Consistency, Clarity)	Interpretability refers to the meaning of terms in the ontology. Clarity is whether the context of terms is clear	The knowledge provided by our ontology is mapping into meaningful real world concepts e.g. book,author,title etc. Also the meaning of each term is clear and consistent.

The instance values stored in database are used by the proposed Ontology based Hidden Web Crawler to fill the Hidden Web form interfaces.This database is also used by the same crawler in proposing a novel technique to map two ontologies. The ontology based hidden web crawler is discussed in next chapter.

Chapter 6

OHWC: ONTOLOGY BASED HIDDEN WEB CRAWLER

6.1 INTRODUCTION

Deep web refers to the contents hidden behind HTML forms. Since it represents a large portion of the structured, unstructured and dynamic data on the web, accessing these types of quality contents has been a long challenge for the database community. This Chapter describes a crawler based on the ontology generated previously for accessing Deep-Web.

Traditional Crawlers provided with a list of URLs picks up a URL called seed URL and downloads the corresponding HTML document. The URL’s embedded therein are appended in to the list of URL’s and the process is repeated. The information which cannot be acquired by simply following hyperlinks and basic keyword searching constitutes hidden web. Hence there is a need of developing a novel architecture of hidden web crawler to extract this information.

In order to download the hidden web contents from the WWW the crawler needs a mechanism for Search Interface interaction i.e. it should be able to download the search interfaces, automatically fill them and submit them to get the hidden web pages. By combining the hidden web retrieval with domain specific ontology, the proposed work automatically fills in the text boxes with values from already defined ontology. This not only makes the retrieval process task specific, but also increases the likelihood of being able to extract just the relevant subset of data.

6.2 PROPOSED ONTOLOGY BASED HIDDEN WEB CRAWLER

The proposed architecture of Ontology based Hidden Web Crawler (OBHWC) consist of following six major components as shown in figure 6.1:

i) Form Downloader
ii) Form Analyzer
iii) Mapping Module
iv) Query Generator
v) Revisit Frequency Calculator
vi) Dispatcher

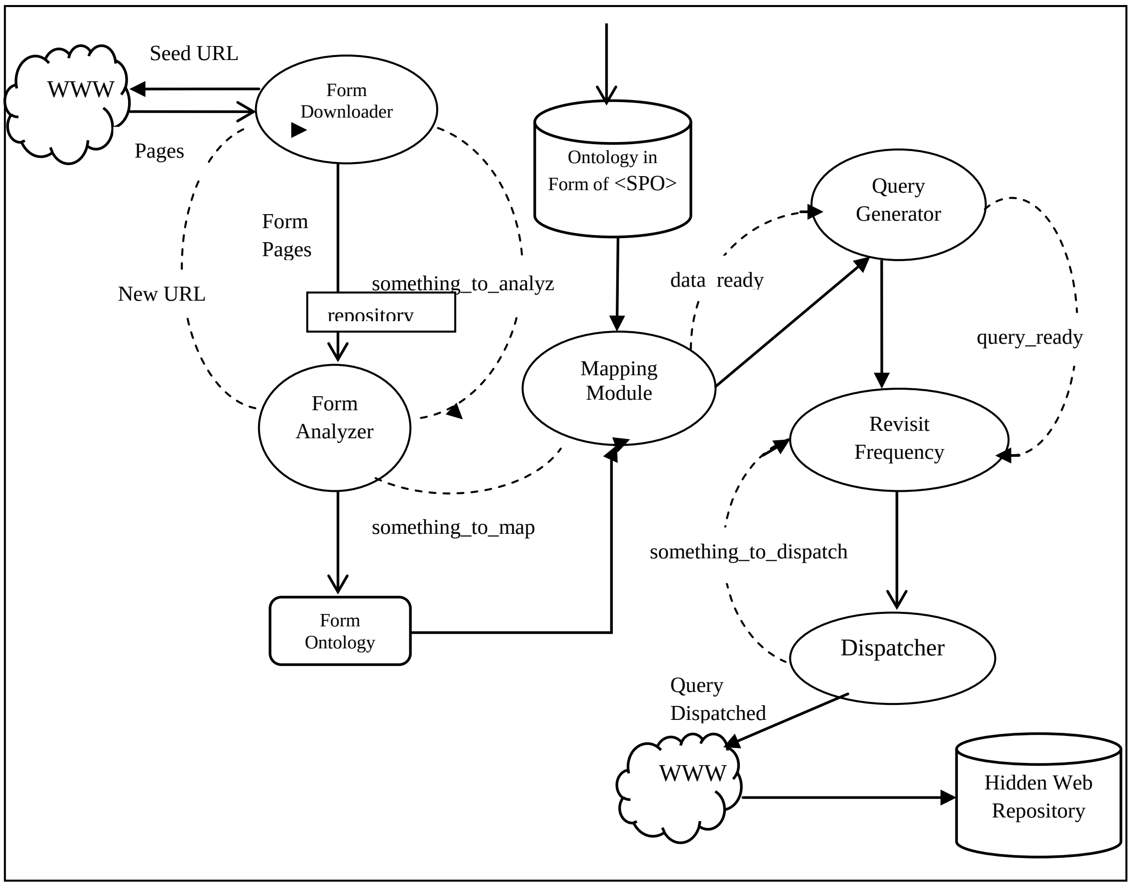


Figure 6.1 Proposed Architecture of Ontology Based Hidden Web Crawler

Brief description of each component is as follows:

i) Form Downloader: The first component is form downloader which is responsible for downloading form pages i.e. only those pages which contain interfaces to be filled by user. These are also specified as entry points to hidden web from WWW. To do this, a seed URL is given to form downloader to start downloading the web pages. Form downloader checks the HTML code of the page and downloads the pages that contain <*form*> tag in them as shown in figure 6.2.

```
Algorithm:
Form_ Downlaoder()
    Step 1: Begin
        2: do forever
        3: download web page corresponding to seed URL.
        4. check if <form> tag is present.
        5. if yes then store in temporary repository
        6. Extract URL’s from downloaded pages
        7. Signal(something_to_analyze);
        8. End
```

Figure 6.2 Algorithm of Form_Downloader

ii) Form Analyzer: This component of Hidden Web Crawler takes the form page from the downloader as input and analyzes the page having different formats. It creates the ontology for the form page using the same algorithm used in PHASE I, (Part 2) to create the ontology from web pages. The ontology created here is also stored in the form of <S,P,O> which is used by next component of OHWC. The algorithm for the same is given in figure 6.3 below.

```
Algorithm:
  Form_ Analyzer()
        Step 1: Begin
             2: do forever
             3: Wait(something_to_analyze);
             4. call ontology_converter();
             5. store form ontology in buffer
             6. Signal(something_to_map);
             7. End
```

Figure 6.3 Algorithm of Form_Analyzer

iii) Mapping Module: This is an important module as the efficiency and output of the OBHWC depends upon accuracy of this function. This module proposes an algorithm to map two different to find exact values that need to be filled on form interfaces. First ontology is the form ontology created by form analyzer above, which is mapped to the domain specific ontology created before in chapter 4. Here semantic matching between two ontologies has been proposed for determining the output mapping function. It means not only two similar words give a matching but also if they are having a relationship like synonyms, siblings, child and parent etc.; have also been considered. This module is explained in detail further with design of a novel mapping technique in section 6.3.

iv) Query Generator: After finding a match by mapping module between two ontologies; queries are being generated for filling the form interfaces taking matched values from domain specific ontology (DSODB). For each matched value a unique query URL is generated here. These queries are then sent to next module to be fired on WWW and obtain the desired result. These queries are temporarily stored in a URL queue which is being accessed by dispatcher module going through revisit frequency calculator. The algorithm for the same is shown in figure 6.4.

It is a post mapping component. It receives a set S of pairs of nodes that are found equivalent during the mapping process of the system. On receiving this set S from Central Coordinator component, it picks one pair at a time and generates the instance/object value associated with particular subject for each mapped record in database. Once these values are generated, a URL is constructed for each particular value corresponding to the format of particular interface. In this way, from each matched pair, a value is computed and placed in the corresponding field. Eventually query URL is submitted by dispatcher through revisit calculator to get the response page and thus enabling retrieval of hidden web.

```
Algorithm Query_Generator(S)            // S comprise matched pair of node
    Step 1: Wait(data_ready);
         2: repeat step 3 to step 5 for each pair of node(n1,n2) in S do.
         3: Generate a value satisfying both domain & range constraint.
         4: Put the value generated in previous step into the query format of form.
         5: if all mandatory fields are filled then goto step  6.
         6: Signal(query_ready);
         7: Call Revisit_Calculator();
         8: Submit Query;
         9: End.
```

Figure 6.4 Algorithm for Query Generator

v) Revisit Frequency Calculator: A URL queue is maintained which contains all the queries generated above and with each URL, a value is associated in queue named revisit frequency. This revisit frequency is calculated according to the domain of each URL so as to save the network bandwidth.

All the queries generated in the form of URL’s from query generator are stored in a URL queue. Before submitting these queries to WWW a revisit frequency is associated to each URL depending upon the domain of the query. There are pages in Hidden Web that contain current information, hence need to be updated continuously e.g. in case of Airline reservation and Railway reservation System the current status is required every time.

In contrast there are web pages that contain factual information e.g. ages that contain history and pages that contain science theories etc. Hence they are not required to be re-crawled frequently. Also there are pages that belong to book domain these websites generally change their data fortnightly or weekly hence they are re-crawled accordingly. For this a value called revisit frequency (rev_fre) is associated and being sent to Dispatcher with each URL/query generated. The algorithm for revisit frequency calculator is shown in figure 6.5.

```
Algorithm Revisit_Calculator()
Step 1: Wait(query_ready);
     2: Find the Domain of the query.
            if  Book domain Set rev_fre to 15 days.
            if Airline domain set rev_fre to 30 mins.
            if factual sites set rev_fre to 30 days.
     3: Update the value of Rev_fre in array of queries.
     4:  Add query to dispatcher() queue.
     5: Signal(something_to_dispatch);
     6: End;
```

Figure 6.5 Algorithm for Revisit Frequency Calculator.

vi) Dispatcher: A URL dispatcher is maintained here which takes the URL from URL queue one by one. Then check each URL's revisit frequency and according to rev_freq if the page is required to be downloaded again (e.g. by checking the last update time/date) then particular query is fired on WWW. The algorithm for the same is given in figure 6.6 below.

The dispatcher here removes one URL from the queue constructed above and checks the revisit_frequency associated with each. Depending upon the value of rev_fre variable value the query is fired on WWW. The resultant Hidden web Pages are then stored in Hidden Web Repository.

```
Dispatcher()
  Step 1: Wait(something_to_dispatch);
       2: While URL queue != null
       3: If new query then fetch URL from WWW.
       4: Else Check rev_fre and last updated date and time form the web page
       5: Depending on rev_fre  dequeue and fetch URL from WWW.
       6: Store the result in Hidden Web Repository.
       7: End.
```

Figure 6.6 Algorithm for Dispatcher

6.3 MAPPING MODULE: Ontology defines common vocabulary for those who need to share information in a domain. Because similar concepts in a specific domain can be represented through different vocabulary there is a requirement to map ontology's (figure 6.8) belonging to same domain in order to make them interoperable. For example, in a person domain both "Man" and "Male" represents similar context (figure 6.7) and it is obvious for human beings but it means different for computers because of different syntax.

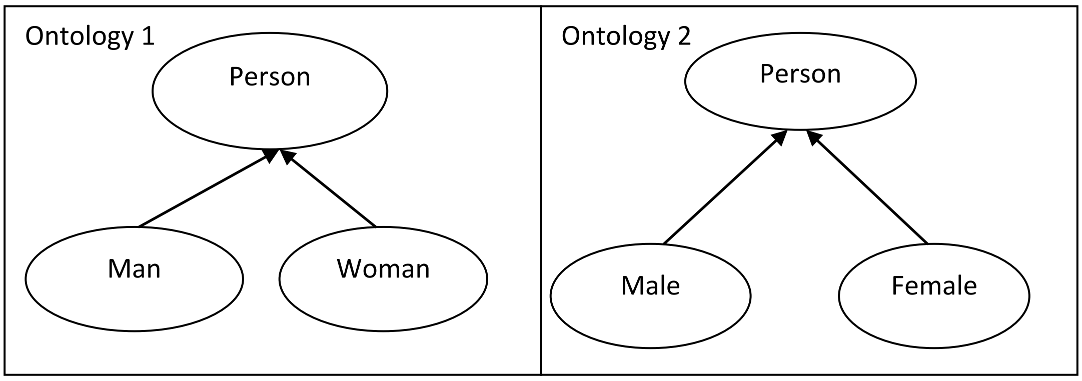


Figure 6.7 Two Ontologies of Same Class Person with Different Vocabularies.

This work focuses on developing an effective and efficient source-based ontology mapping technique for a specific domain, which finds the appropriate values that need to be filled in the form interfaces.

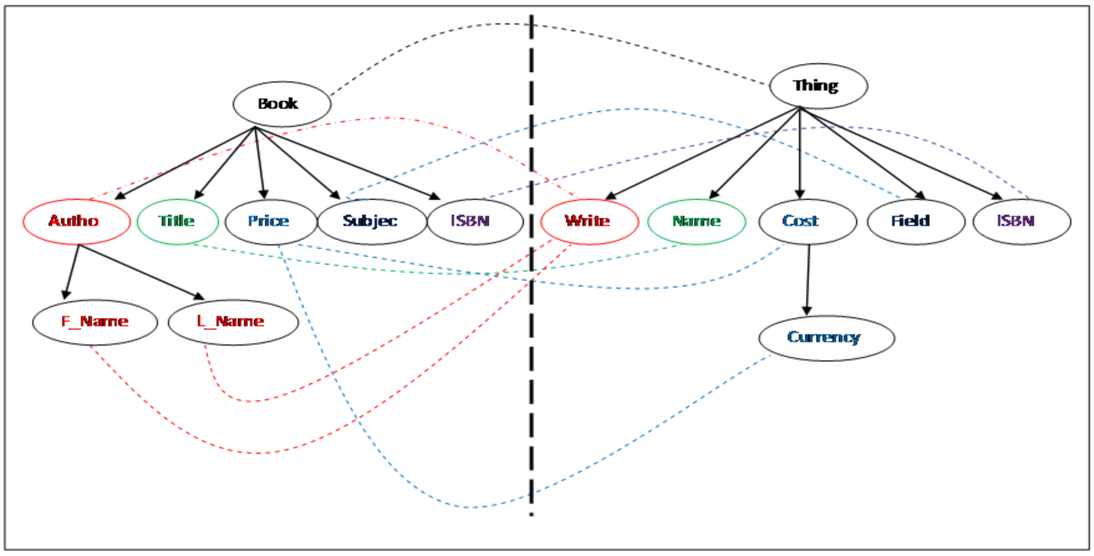


Figure 6.8 Diagram Showing Mapping of Two Book Domain Ontology

Here in this module the form's ontology is mapped to the domain specific ontology generated before. For each mapping concept pair the values from predefined ontology are extracted and filled in the interface. The primary focus is to make the process more efficient in terms of resources consumed during the process. The complexity of matching process usually varies in accordance with the size of the ontology under consideration.

A straightforward approach for ontology mapping is to reduce the number of pair-wise comparisons. Out of so many possible concept-pairs only a few are considered for mapping based on the depth of the query-form ontology. These concept-pairs are further reduced to a minimal set by eliminating irrelevant pairs. Once a minimal set of concept pairs is obtained, mapping between these pairs are performed in two steps: First is linguistic mapping and second is structure mapping. The mapping technique mainly consists of following components and the architecture of this novel technique is shown in figure 6.9 above.

i) Central Coordinator
ii) Pre-Processor
iii) Main Mapper

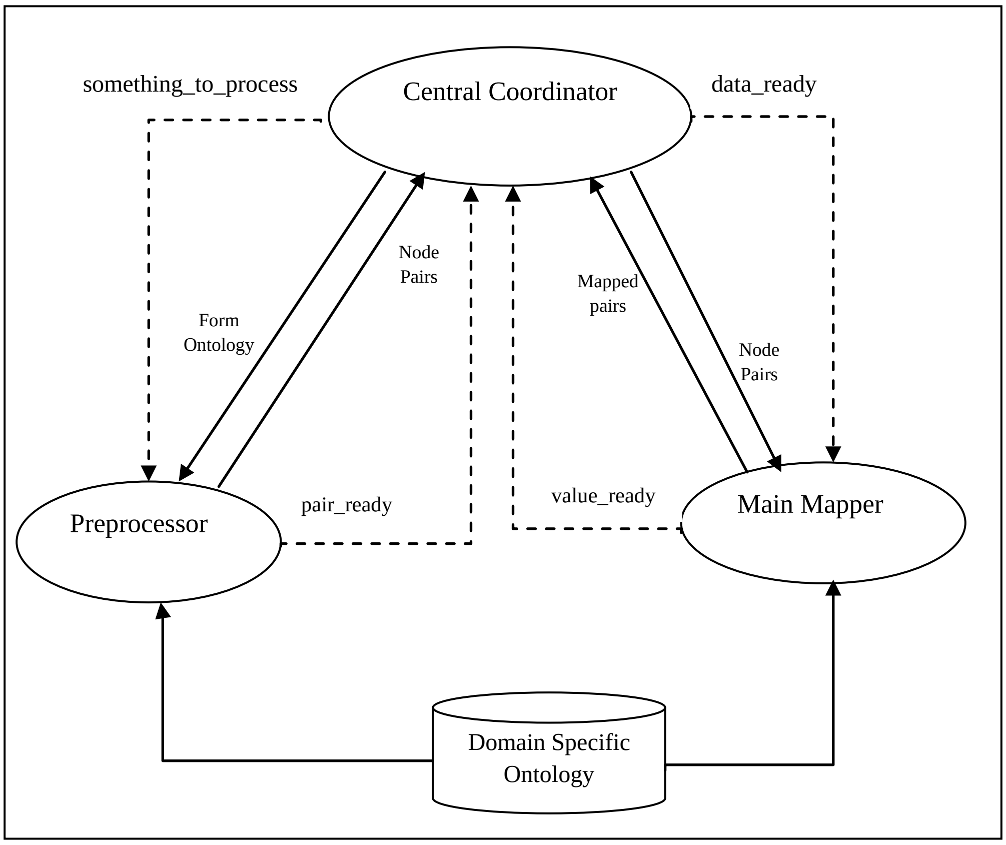


Figure 6.9 Proposed Architecture of Ontology Mapping System

The detailed description of each sub component is as follows:

6.3.1 Central Coordinator: Central coordinator is the main processing component of the proposed technique and it coordinates rest of the major components of the system. It provides input to all main components and receives output from them. On receiving output from one sub ordinate component, Central Coordinator module pass this information to the next major subordinate module in the system and repeat the same until the work is done. The algorithm for the central coordinator is shown below in figure 6.10.

Central Coordinator provides form ontology as input to the Pre-processor component which returns back a set of pairs of nodes as output. Output received from pre-processor acts as input to the main-mapper component and output generated by main-mapper is provided to central coordinator which transfers it to the query generator component of OHWC which finally ends with a response page corresponding to a query interface form.

```
Central Coordinator ( )
      Step 1: Begin
              2. do forever
              3. Wait(something_to_map);
              4. Signal(something_to_process);
              5. Call Pre-Processor;
              6. Wait(pair_ready);
              7. Signal(data_ready);
              8. Call Main_Mapper();
              9. Wait(value_ready);
             10.Signal(query_ready);
             11. Call Query_Generator;
             12. End
```

Figure 6.10 The Algorithm for Central Coordinator Module

6.3.2 Pre-Processor: This module operates on two inputs provided by the coordinating module. It encompasses three sub ordinate modules as shown in figure 6.11. The name of the subcomponents is:

i) Node Selector

ii) Random Filler

iii) Pair Constructor

One of the input is form interface in form of form ontology and the second is ontology database (of same domain) is taken for reference ontology. From these two inputs this module makes pairs of concepts (nodes) from both ontology for which mapping has to be performed for equivalency. Prior to making pairs of concepts a component called random-Filler is employed that randomly fills-in values in the query-form and submit it. Subsequently, an error report is generated by Error-handler module, specifying mismatch of data types if any. Once this knowledge is obtained regarding the data types of the fields (concepts) in query-form, only concepts in ontology database with similar data type are paired and thus eliminating the pairing of irrelevant concepts from the both ontology. After making a set of pairs of nodes from both ontologies, it returns this set to the Central Coordinating module for further processing.

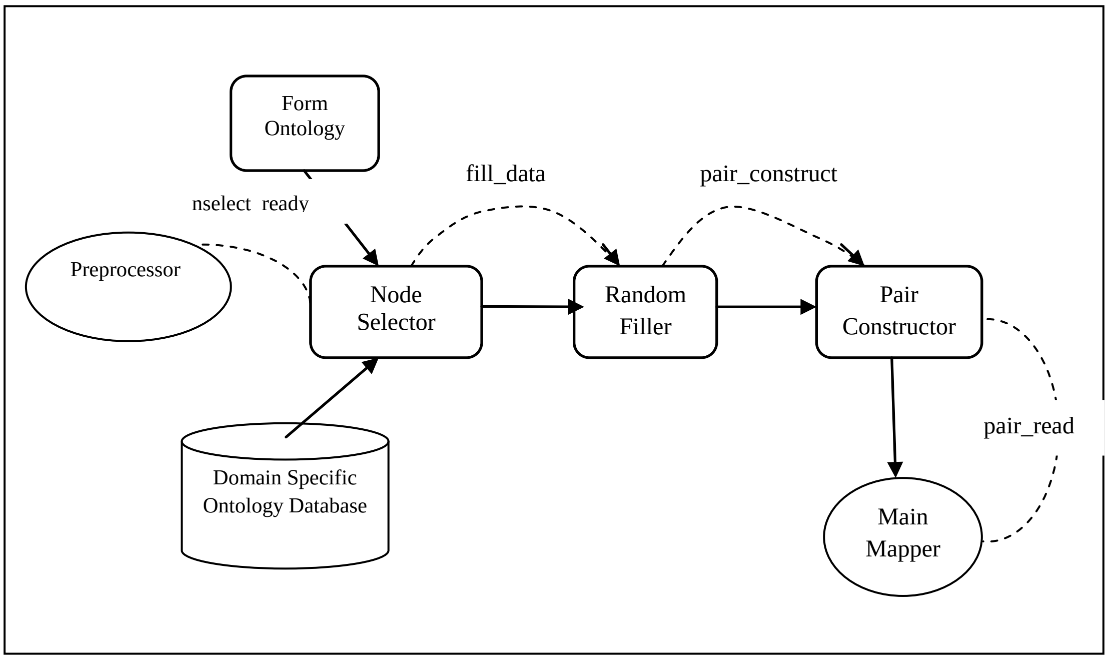


Figure 6.11 Pre_Processor's Sub-Components

The detailed description of each subcomponent along with their algorithms is given in next section. The algorithm for pre-processor module is described in figure 6.12 below.

```
Pre_Processor ($O_1$, $O_2$)
    Step 1: Wait(something_to_process);
        2: Signal(nselect_ready);
        3: Signal(fill_data);
        4: Get data type mismatch form error report.
        5: Signal(pair_construct);
        6: return set of pair of nodes to central coordinator.
        7: Signal(pair_ready);
        8: End
```

Figure 6.12 Algorithm for Pre processor Module

i) Node selector: This module receives as input form's ontology which is represented in the form of graph. As an output it produces some or all nodes of ontology according to the levels of nodes in the graph. Output of this process is then provided to Random Filler module for further processing. The algorithm for the same is shown in figure 6.13.

```
Node_Selector (G, D)//G is a graph corresponding to ontology & D is a depth.
        Step 1: Wait(nselect_ready);
             2: create and initialize a set S with root
             3: insert root into an initially empty queue Q.
             4: repeat step 4 to 8 while Q ≠ Φ and L ≠ 0
             5: extract a node u from Q
             6: repeat step 7 & 8 for each node v∈ Adj[u] do
             7: add v into S
             8: reduce L by one.
             9: return S.
             10: Signal(fill_data);
             11: End.
```

Figure 6.13 Algorithm for Node Selector Subcomponent

ii) Random Filler: This component is developed to increase efficiency. In a pair wise mapping major of the time complexity comes from the number of mapping pairs. Keeping in mind this fact, only relevant pairs (having larger possibility of matching) are considered for mapping. Random Generator deals with this task. For each mandatory concept in the ontology1 a random value is generated and filled-in. After submitting the query, usually an error report is produced by the interface describing the data type discrepancy, if any. This report is supplied to pair Constructor module. The algorithm of random filler subcomponent is shown in figure 6.14.

```
Random_Filler(O1)
        Step1: Wait(fill_data);
             2: Repeat step 3 & 4 for each concept C in ontology O1.
             3: Value = Random ();      //function that generates a random number
             4: Fill-in the value into the field of C.
             5: Submit the form
             6: Receive error report if generated
             7: Analyze report for data type  mismatch error.
             8: Return the data type of nodes to pair constructor.
             9: Signal(pair_construct);
            10: End
```

Figure 6.14 Algorithm for Random_Filler

iii) Pair Constructor: It receives set of nodes S1 and S2 and a error report. This discrepancy report describes appropriate data type for each node. For the nodes that have similar data type pairing is done discarding mismatched data type values. For example: integer fields should pair only with integer fields and string fields should pair only with string fields. From these two sets S1 and S2 it generates another set S (pair set) which includes Cartesian products of these two sets considering discrepancy report. The algorithm of pair constructor is described in figure 6.15. Every pair contains one node from each set that has to be matched.

$S_1 = \{a_1, a_2, .., a_n\}$ and $S_2 = \{b_1, b_2, ..,b_n\}$

$S = \{ (a_1,b_1), (a_1,b_2), \ldots, (a_1,b_n), (a_2,b_1), (a_2,b_2),\ldots, (a_2,b_n), .., (a_n,b_n)\}$

In general $S = (a_i, b_j)$ Where $a_i \in S_1$ and $b_j \in S_2$

The set of pairs of node are returned as output to the Central Coordinator module for actual mapping between them.

```
Pair_Constructor ($S_1$, $S_2$, DR)
    Step 1: Wait (pair_construct);
         2: Make an empty set S
         3: Repeat step 4 for each node $n_1 \in S_1$
         4: Repeat step 5-7 for each node $n_2 \in S_2$
         5: If pair ($n_1$, $n_2$) both shares same data type, than goto step 7.
         6: goto step 4.
         7: Add pair ($n_1$, $n_2$) into set S.
         8: Return S.
         9: Signal (pair_ready);
         10: End.
```

Figure 6.15 Algorithm for Pair_Constructor

6.3.3 Main Mapper: The task of Main Mapping is broadly divided into two phases, Linguistic matching and Structure matching. In linguistic matching name of the concepts (nodes) are compared with some existing vocabulary (synonyms) whereas during structure matching structure of the underlying nodes are examined. Main Mapper receives input from central Coordinator module after the initial pre_processing. The input received by it is a set S of pairs of nodes whose Linguistic Matching & Structure matching is verified here for equivalency and thus it is responsible for actual mapping task.

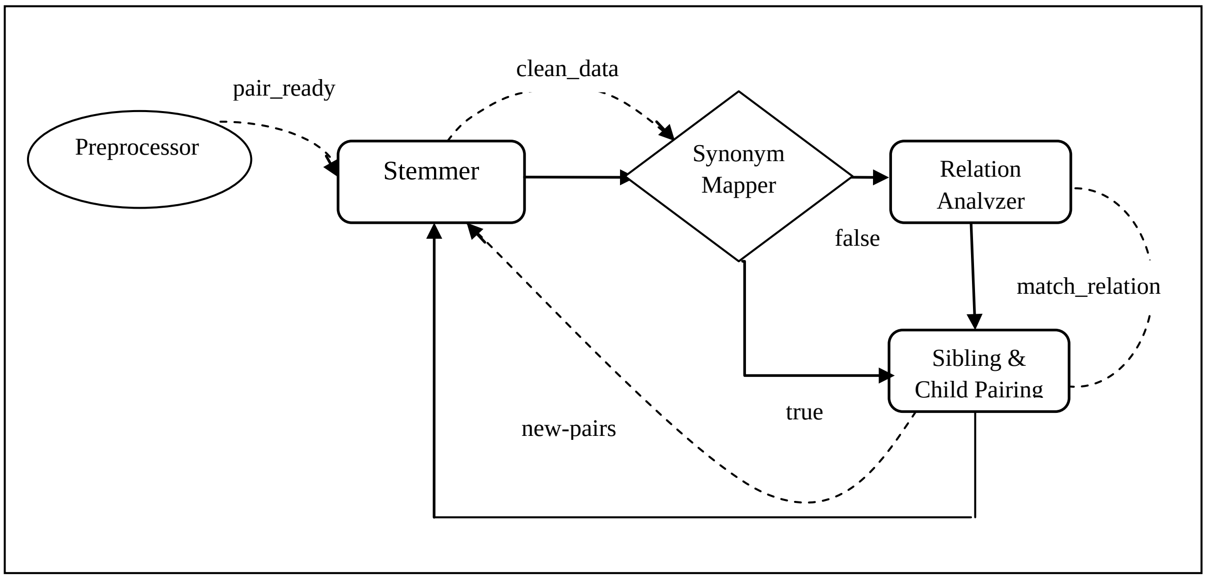


Figure 6.16 Architecture of Main Mapper Component

This module consists of four sub ordinate module to do the above mentioned task a shown in figure 6.16.

i) Stemmer

ii) Synonym Mapper

iii) Relation Analyzer

iv) Sibling & Child Pairing

Main_Mapper(S)

```
Step  1: Wait (pair_ready)
      2:  Repeat step 3 to 10  for each pair of node ($n_1$,$n_2$)from S.
      3:  Call Stemmer ({$n_1$, $n_2$});
      4:  Signal (clean_data);
      5: Call Synonym_Mapper ({$n_1$, $n_2$}) which returns either true or false depending on
          whether both nodes $n_1$ & $n_2$ are found equivalent or not by performing linguistic
          matching.
      6:  if step 5 returns false then goto step 8
      7:  Call Sibling&Child_Pair ($n_1$, $n_2$) which makes some additional pair of child &
          sibling of pair($n_1$, $n_2$) and goto step 11
      8: Call Relation_Analyzer ($n_1$, $n_2$) which returns either True or False by performing
          structure matching
      9: if step 8 returns false then goto step 12
     10: goto step 7
     11: Continue.
     12: Signal(value_ready);
     13: End
```

Figure 6.17 Algorithm of Main Mapper

Main mapper's algorithm is shown in figure 6.17. It is responsible for doing these below specified tasks:

i) It is responsible for converting the concept (field name) into its base/stem form.

ii) It involves the process of checking whether concept from ontology 2 contains concept of ontology 1 as its synonym in Ontology database or not and returns true or false respectively.

iii) If synonym is not found in the Ontology database than structure of both concepts from the pair are examined. True value is returned if structure is similar else false is returned.

iv) Finally, if pair is found to be equivalent (linguistically or structurally) than their sibling and child are also paired for the same evaluation.

Detailed descriptions of its subordinate components are mentioned here:

i) Stemmer: It is a subordinate module of Main Mapper module. Stemmer module is invoked just before performing linguistic matching. It involves the process for reducing inflected (or sometimes derived) words to their stem, base or root form—generally a written word form. All concepts from query interface ontology (Ontology1) are goes through this component and their corresponding stems are obtained. A temporary data structure records each concept from Ontology1 and their corresponding stem. The data structure created by this process is utilized during linguistic matching.

ii) Synonym Mapper: This component takes first input from stemmer module & second input from centralized domain specific database. Synonym mapper checks out whether the nodes in the pair $S(n_1,n_2)$ are synonyms or not. The algorithm for the same is given in figure 6.18. For this it matches the stemmed values of n_1 in the synonym table for n_2 from DSDB. If an entry is found it means a match has occurred and this pair of concepts are added to the database called Mapping_Vector. In this case sibling & child of this equivalent pair need to be matched. To do so recently matched pair is supplied to Sibling & Child Pairing component. If pair is not found to be matched than they are passed to relation analyzer component for Semantic mapping. If pair is not found to be matched than they are passed to relation analyzer component for structure matching.

```
Synonym_Mapper(n1, n2)
    Step 1: Wait(clean_data);
         2: Receive a location loc generated by a hash function by passing n1 to
            this hash function.
         3: Repeat step 4 to 8 while loc ≠ Nil
         4: If information at loc is matched with that of n2 goto step 6
         5: Let loc now points to next node in the list
         6: Increment matched by one.
         7: Add this pair to Mapping_Vector by setting word1 = n1 & word2 =
            n2       // Mapping_Vector is a database containing matched concepts
         8: Return true.
         9: Return false.
```

Figure 6.18 Algorithm for Synonym Mapper

iii) Relation Analyser: For any pair of node this module is invoked if there is no linguistic matching found for them and structure matching is required for next level examination. Upon receiving a pair of nodes this module performs a various quality tasks.

```
Relation_Analyser($n_1$, $n_2$)
    Step 1: If relation of node $n_1$ with its parent is not same as that of $n_2$ then goto step 3
         2: Assign $W_1=\alpha_1$
         3: If cardinality of $n_1$ is not same as that of $n_2$ goto step5
         4: Assign $W_2=\alpha_2$
         5: Repeat step 6 and 7 for each type of relation
         6: Check whether no of relation in n1 is almost same as that of $n_2$ goto step 8
         7: Goto step 9.
         8: Assign $W_3=\alpha_3$
         9: Compare $W = W_1 + W_2 + W_3$ with a predefined threshold Th.
        10: If W is qualified then goto step 13
        11: Increment mismatch by one.
        12: Return false
        13: Add this pair to Mapping_Vector by setting $word_1 = n_1$ & $word_2 = n_2$
        14: Signal(match_realtion);
        15: Return true
```

Figure 6.19 Algorithm for Relation_Analyser

First of all it checks whether these two nodes are connected with their parent with the same relation or not. If the relation with which they are connected with their parents are found to be same then a specific weight is assigned to this pair. After done with this step, the no of relations as well as their types with which these two nodes are associated are analyzed. If number of relations for each type is almost same for both pair then a predefined weight is assigned to this pair.

Eventually the weight associated with each pair is compared with a threshold value. If this weight is qualified for the threshold then that pair of node is considered as equivalent and this pair is added to the Mapping-Vector (database), if the weight is not qualified for threshold then this pair is discarded. The algorithm for relation analyser is given in figure 6.19

iv) Sibling & Child Pairing: This module is invoked only when a match is found between a pair of nodes. So, this component is connected with both Linguistic Matcher & Structure Matcher. After receiving a node pair it calls Sibling Mapper & Child Mapper that makes pairs of their sibling & child respectively. Since for every new pair their parents are equivalent, there is a possibility that they too can be similar, so by this assumption a predefined weight is assigned to them. Once pairing of all possible siblings & child's has been formed it invokes sstemmer module for the next pair in the queue. The algorithm for sibling & child pairing is given in figure 6.20 below.

```
Algorithm
Sibling&Child_Pairing ( $n_1$, $n_2$)
    Step 1: Call Sibling_Pair($n_1$, $n_2$)
         2: Call child_Pair($n_1$, $n_2$)
         3: End
Sibling_Pair($n_1$, $n_2$)
      Step 1: Set p1 = $\pi[n_1]$.
           2: Set p2 = $\pi[n_2]$.
           3: Repeat step 4 for each node $u_1 \in$ Adj[p1] and $u_1 \neq n_1$ do
           4: Repeat step 5 and 6 for each node $u_2 \in$ Adj[p2] and $u_2 \neq n_2$ do
           5: Add pair($u_1$, $u_2$) into set S.
           6: Increment Total by one. // Total contains total no of matched
              concept-pairs.
           7: End.
Child_Pair($n_1$, $n_2$)
        Step 1: Repeat step 2  for each node $u_1 \in$ Adj[$n_1$] and $u_1 \neq n_1$ do
             2: Repeat step 3 and 4 for each node $u_2 \in$ Adj[$n_2$] and $u_2 \neq n_2$ do
             3: Add pair($u_1$, $u_2$) into set S.
             4: Increment Total by one.
             5: End.
```

Figure 6.20 Algorithm for Sibling & Child Pairing

6.4 PERFORMANCE EVALUATION OF MAPPING MODULE

Evaluation of quality of matched result of the proposed system is based on the Compliance Measures. Compliance Measures are used to evaluate the degree of compliance of the results of matching algorithms. Compliance measures consist of three measures Precision, Recall and F-measure. These three matrices are employed to measure the performance of concepts (fields) mapping.

Precision is a measure of the number of correct mappings found versus the total number of retrieved mappings (correct and wrong).

Recall describes the number of correct mappings found in comparison to the total number of existing mappings. Suppose the number of correctly identified mappings is C, the number of wrongly identified mappings is W and the number of unidentified correct mappings is M. then the precision of the approach is given by the expression given below:

$$P = C / (C + W) \qquad \text{(Eq. 6.1)}$$

and recall is formulized as:

$$R = C / (C + M) \qquad \text{(Eq. 6.2)}$$

The f-measure combines the two parameters precision and recall as a single efficiency measure. The standard formula is defined as:

$$F = ((b^2+1)*P*R) / (b^2*P+R) \qquad \text{(Eq. 6.3)}$$

b is a factor used to quantify the value of precision and recall against each other.
For the consequent test runs b is considered 1. Thus taking b = 1, F-measure can be given as:

$$F = 2PR/ (P + R) \qquad \text{(Eq. 6.4)}$$

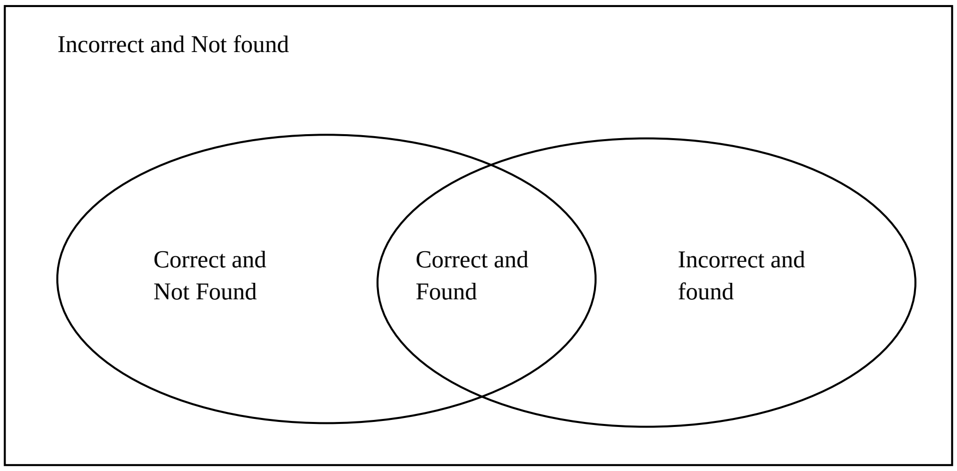


Figure 6.21 Venn Diagram Illustrating Sample Evaluation

As mentioned in the above figure 6.21 in the Venn diagram, concept-pairs are classified into four categories:

a. Correct and found: Pairs that has been identified and found as correct.
b. Correct and not found: Pairs that are not identified yet, but believed to be correct.
c. Incorrect and found: Pairs which are identified but found to be incorrect.
d. Incorrect and not found: Pairs neither identified nor believed to be correct.

With these classification concept-pairs from ontology1 & ontology2 as represented in figure 6.3 are categorised as follows:

Correct and found (a) = {(First name, Writer), (Last name, Writer), (Price, Cost), (ISBN, ISBN)}

Incorrect and found (b) = {(First name, Name), (First name, Field), (Last name, Name), (Last name, Field), (Title, Writer), (Title, Field), (Price, Writer), (Price, Name), (Price, Field), (ISBN, Cost), (Price, ISBN)

Correct and not found(c) = {(Subject, Field), (Title, Name)}

Incorrect and not found (d) = {(Subject, Writer), (Subject, Name)}

i) Comparison: For the purpose of evaluating this work, another mapping system called traditional mapping system is considered. This traditional system is based on some existing mapping system that matches all fields (concepts) from first ontology to the all fields of second ontology. The only objective of this traditional ontology mapping system is to compare it with the proposed ontology mapping system.

With the above mentioned classified concept pair's compliance measures are evaluated for both proposed and traditional system. For the traditional system correct & found (a) and correct & not found (c) is same as that of proposed system. In the proposed model only relevant pairs with similar data types are considered for mapping, so there are relatively less concept-pairs as compare to traditional system which comprises all possible concept-pairs from both ontology. All identified but incorrect concept-pairs are referred as b. A table containing computation of performance measures are mentioned in table 6.1.

Table 6.1: Performance of Proposed and Traditional Model

Threshold	**a**	**$b_{proposed}$**	**$b_{traditional}$**	**c**	**$P_{Proposed}$**	**$P_{traditional}$**	**$R_{proposed}$**	**$R_{traditional}$**
0.50	4	11	23	2	4/15=0.27	4/27=0.14	4/6=0.67	4/6=0.67
0.70	3	12	24	2	3/15=0.20	3/27=0.12	3/4=0.75	2/4= 0.5
0.85	2	13	25	2	2/15=0.13	2/27=0.07	2/3=0.67	1/3=0.33
1.00	1	14	26	2	1/15=0.06	1/27=0.03	1/1=1	1/1=1

Figures 6.22, 6.23 and 6.24, present the average compliance measures results (Precision, Recall and F-measure) of all the matched pairs of Book-domain represented in figure 5.3. Result is obtained (manually) at different (0.5, 0.7, 0.85 and 1.0) threshold values.

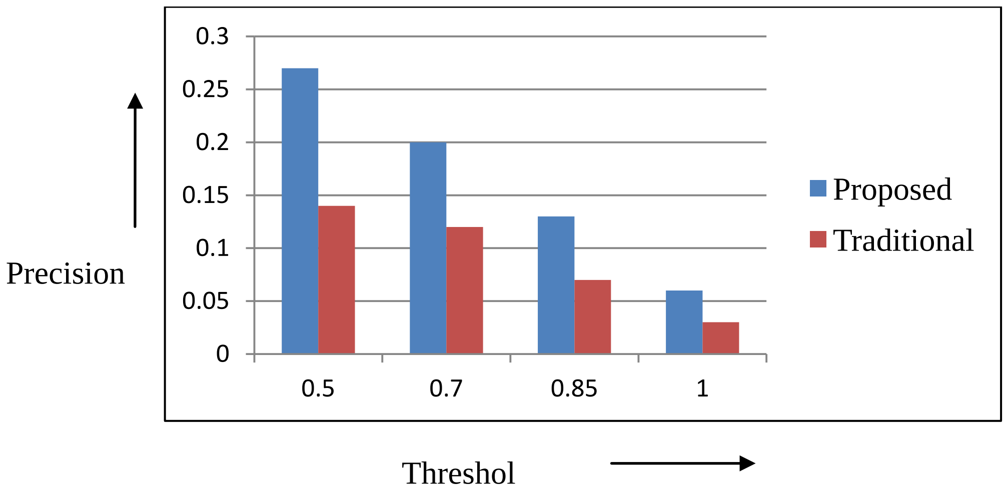


Figure 6.22 Precision of Proposed & Traditional System at Various Thresholds

As clearly depicted from the above graph, precision of the proposed system is always found to be better than that of the traditional system. Hence the proposed system can assumed to be more accurate than the existing one and at the same time it also returns result more efficiently. Precision is highest at threshold 0.5 which implies that system returns largest number of matched concepts. As threshold value is getting increased precision decreases because less matched concepts are found at higher threshold. Similarly recall of the proposed system is compared with that of traditional system as given:

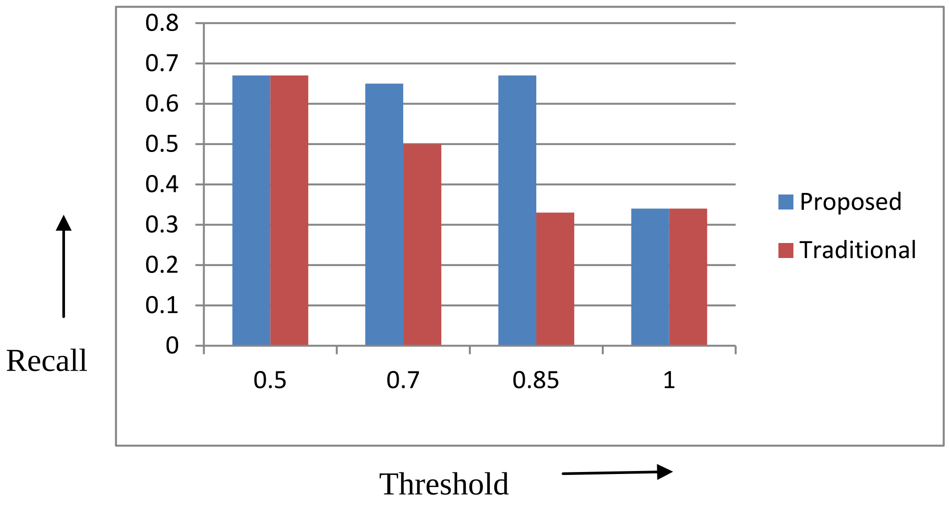


Figure 6.23 Recall of Proposed System & Traditional System at Various Threshold

Since Recall depends on the correct concept-pairs either found or not found and does not rely on incorrect pair (found or not found), hence there is no difference in the result for proposed and traditional model, as proposed model only eliminates irrelevant concept-pair which eventually found to be incorrect.

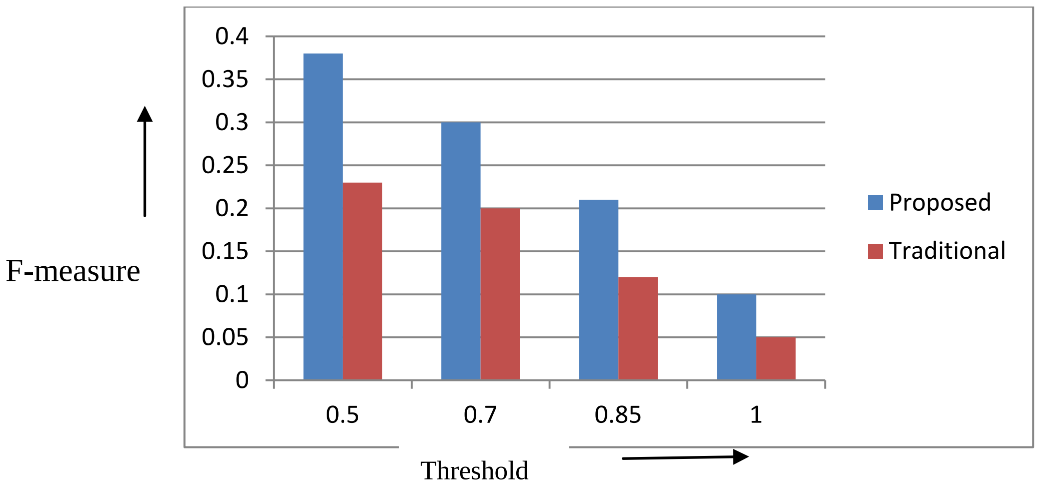


Figure 6.24 F-Measure of Proposed and Traditional System at Different Threshold

As depicted above, at all thresholds (0.5, 0.7, 0.85, 1.0) F-measure of proposed system is always better than the F-measure of the traditional system. This reflects the fact that the proposed system returns more accurate results in terms of total matched concept and total correctly found concepts than the traditional system. In other words, with the proposed system more correctly identified concepts are found as compared to the traditional system.

6.5 PERFORMANCE EVALUATION OF ONTOLOGY BASED HIDDEN WEB CRAWLER

The performance of OHWC is also measured via three metrics: Precision, Recall, and F-measure, Let us define the following:

- number of correctly crawled hidden web pages are C,
- the number of wrongly crawled hidden web pages are W and
- the number of hidden web pages that are not crawled by OHWC are H.

With the help of above defined terms C, W and H, let us now further define the various performance metrics:

1. Precision is defined as a fraction of correctly crawled hidden web pages over all the web pages crawled by the OHWC. Mathematically, the Precision is given by

$$\boxed{P = C / (C + W) \qquad \text{(Eq. 6.5)}}$$

2. Recall is defined as a fraction of correctly crawled hidden web pages by the OHWC over all the hidden web pages as given by domain experts., then the Recall of the proposed system is given by the expression given below

$$\boxed{R = C/ (C + H) \qquad \text{(Eq. 6.6)}}$$

3. F-measure incorporates both precision and recall. F-measure is given by

$$\boxed{F = 2PR/ (P + R) \qquad \text{(Eq. 6.7)}}$$

where Precision P and Recall R are equally weighted.

For experimental evaluation of the proposed work, the two domains Books and Airlines have been considered.

Table 6.2 Table Showing Evaluation Data of Precision Recall and F-measures for Book and Airline domain

Domain	Book	Airline
Number of Websites on which queries are fired	5	4
Total Number of Queries Constructed	189	167
Number of Relevant Results Retrieved	178	158
Number of Error Pages	5	4
Number of Irrelevant Pages Retrieved	6	6
Total Number of Results Retrieved	184	163
Precision	96.7%	96.9%
Recall	94.1%	94.6%
F-Measure	97.6%	95.5%

A. Result Analysis for Book Domain

- **Precision** : Calculated using formula in equation 6.5 = $\frac{178}{184}$ = 96.7% (*100)
- **Recall :** Calculated using formula in equation 6.6 = $\frac{178}{189}$ = 94.1%
- **F-measure** = Calculated using formula in equation 6.7 = $\frac{63368}{64902}$ = 97.6%

Result Analysis for Airline Domain

- **Precision** = $\frac{158}{163}$ = 96.9% (*100)

- **Recall** $= \frac{158}{167} = 94.6\%$
- **F-measure** $= \frac{49928}{52140} = 95.5\%$

Figures 6.25, 6.26 and 6.27 show comparison graphs of Precision, Recall and F-measures for both Book and Airline Domains respectively. The comparison is done between the proposed OHWC and traditional DSHWC.

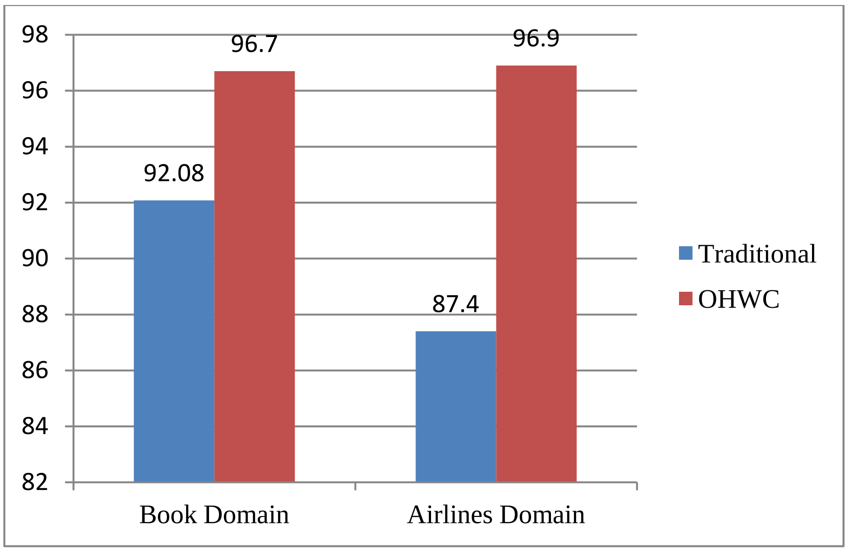


Figure 6.25 Comparison Graph of Precision between Traditional Crawler and OHWC

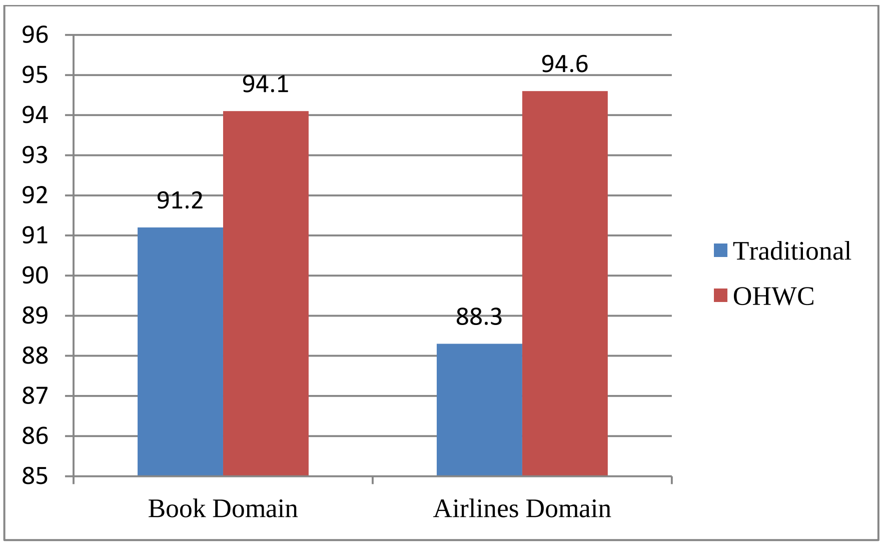


Figure 6.26 Comparison Graph of Recall between Traditional Crawler and OHWC

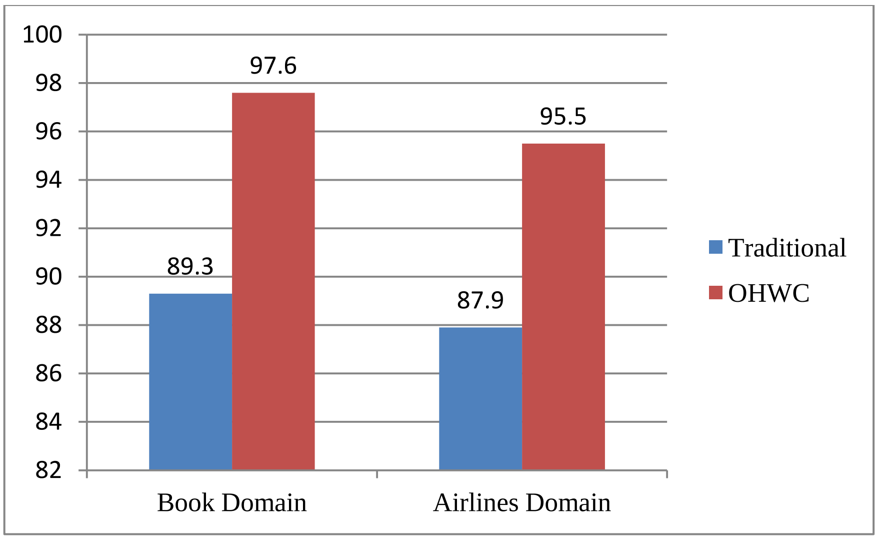


Figure 6.27 Comparison Graph of F-measure between Traditional Crawler and OHWC

From the results it can be observed that there is improvement in the percentage of correct pages. Which in turn helps in improving all three compliance measures Precision, Recall, and F-measure in comparison with traditional HW crawler (DSHWC).

The Ontology based Hidden Web Crawler discussed in this chapter is used as an effective tool to crawl large amount of high quality information hidden behind search forms. It automates the downloading of search interfaces, finds the semantic mappings between them using ontology and finally, the crawler submits the form to obtain the response pages from the Hidden Web.

The hidden web pages downloaded by OHWC are indexed using proposed ontology based indexing scheme. A novel ranking technique is applied on them to present the same to user in an effective way. Both the proposed techniques are discussed in next chapter.

Chapter 7

ONTOLOGY BASED INDEXING TECHNIQUE AND NOVEL RANKING TECHNIQUE FOR HIDDEN WEB PAGES

7.1 INTRODUCTION

Indexing is a way to organise the information of a document so that the same may be retrieved by the end user more efficiently. Without an index, the search engine would scan every document available in the corpus, which would require considerable time and computing power. So granting efficient and fast accesses to the index is a major issue for performances of Web Search Engines.

The indexing process takes the detailed information collected by the crawler and analyses it. The index built by the indexer is based upon the terms of the document. It consist of an array of the posting lists where each posting list is associated with a term and the identifiers of the document containing the term as shown in figure 7.1.

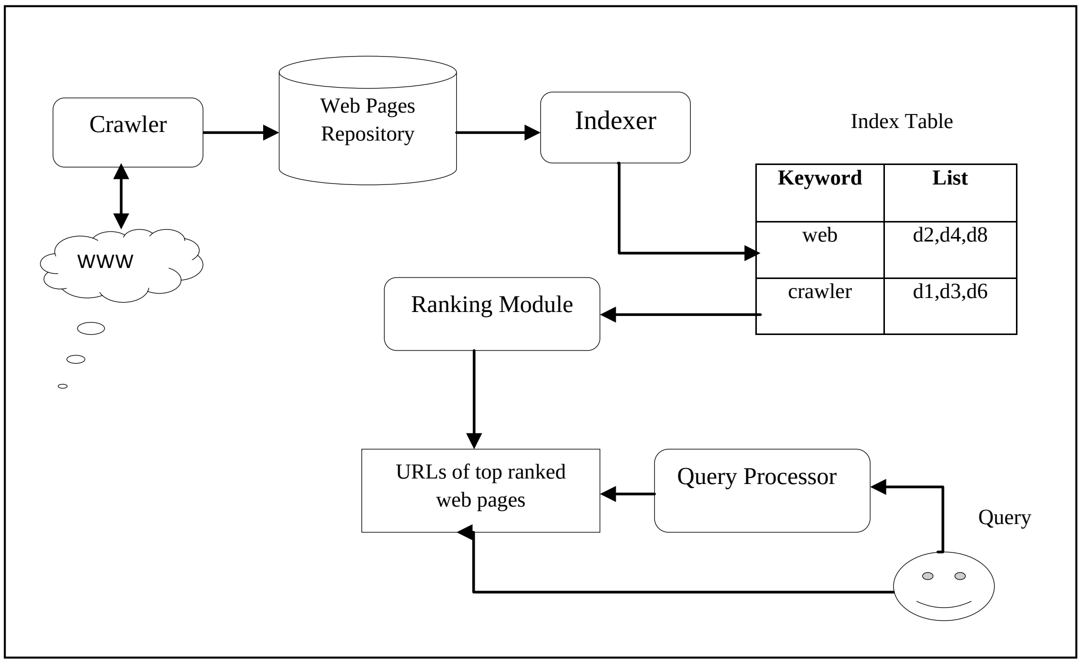


Figure 7.1 Basic Architecture of Search Engine Showing Indexing and Ranking of Documents.

The current information retrieval systems use terms to describe documents and search engines consider keyword frequency as a major factor to determine its association with a page and does not take into account the context (sense) in which that keyword appears in document. By using ontology the concept with a term can be attached. But having attached the concept only does not solve the whole problem, as there may be some ambiguous terms also. Therefore a new ontology based indexing sheme is proposed and implemented in this chapter.

After indexing user gets the URL's of different web pages depending on the query entered. A general query engine returns a number of pages that match the keywords for a given query. But all of them may not be relevant and user has to scroll down to next page for getting desired information. Therefore to provide better search result, page ranking mechanisms are used by most search engines for putting the important pages on top leaving the less important pages in the bottom of result list. There are various page ranking algorithms for surface web. But there is a scarcity of ranking algorithms for hidden web pages. In this chapter a novel efficient ranking technique for Hidden web data is also discussed.

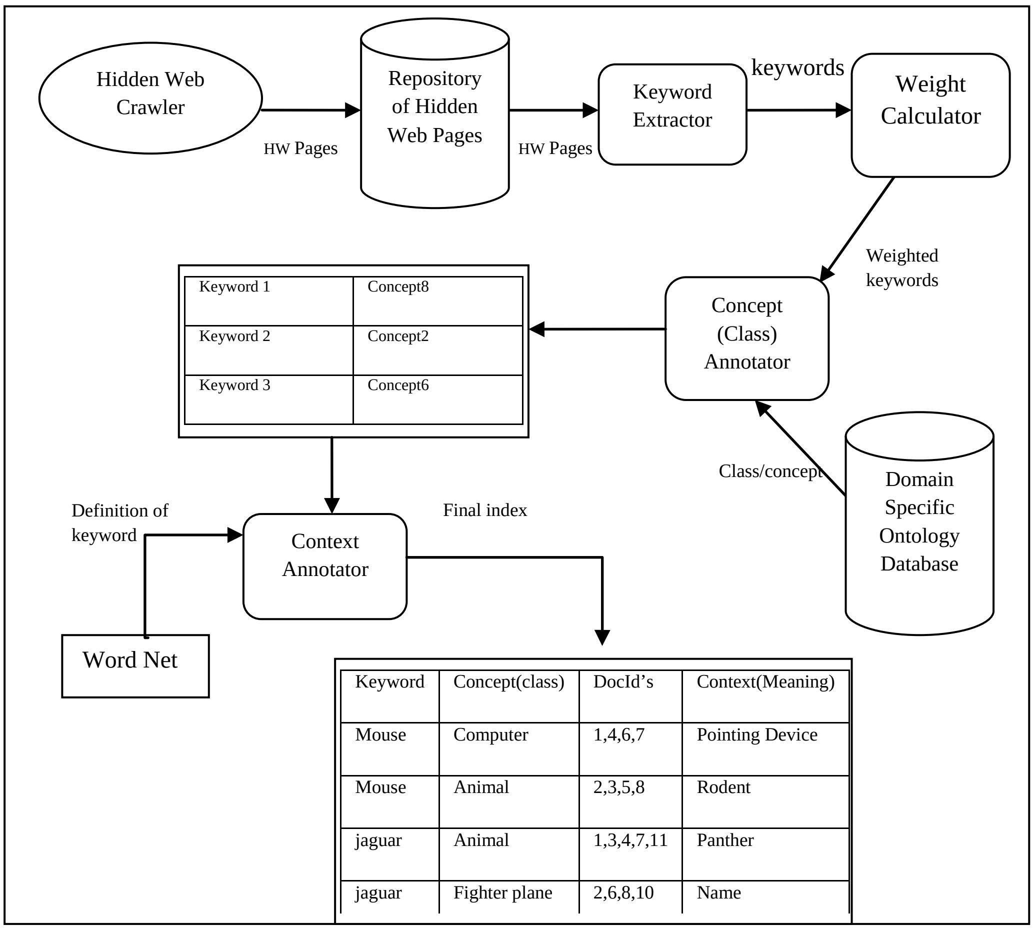
7.2 PROPOSED ONTOLOGY BASED INDEXING TECHNIQUE

The proposed ontology based web indexing system attaches different class (concept) to the keyword and does a mapping between keyword and ontology class. For resolving ambiguities the proposed system uses WORDNET to attach the context with every keyword in which it is used is attached. Due to attachment of context with keyword, more relevant results are retrieved in optimized way in lesser time response to a search query.

The architecture of a new ontology based indexing technique, for the result set retrieved by hidden web crawler is shown in figure 7.2 below. The components of the proposed architecture are:

i) Keyword Extractor
ii) Weight Calculator
iii) Concept Annotator
iv) Context Annotator

The proposed model takes the domain specific pages to create index. A specific domain is chosen because the ontology which is used is made for particular domain. Two domains have been chosen, explored and ontology has been developed for them. The ontology used here has already been constructed in chapter 3, 4 [74, 76].



Keyword	Concept(class)	DocId's	Context(Meaning)
Mouse	Computer	1,4,6,7	Pointing Device
Mouse	Animal	2,3,5,8	Rodent
jaguar	Animal	1,3,4,7,11	Panther
jaguar	Fighter plane	2,6,8,10	Name

Figure 7.2 The Proposed Architecture of Ontology Based Indexing Technique for Hidden Web.

Now the previously downloaded web pages by OHCW are indexed word by word. In first step, normal keyword based indexing is done. This results in a cleaned up wordlist for each document, which is stored in a table. Now the keyword to be indexed is mapped to a Concept in ontology. Further to remove the ambiguities in concepts the context in which a word is used is also stored in index table.

The detailed description of each component along with illustration is given below:

7.2.1 Keyword Extractor: This component parses the entire web page and extracts all keywords from the text of the document, discarding stop words. After getting all keywords, stemming is applied to get the pure form of each keyword. These keywords are sent further to weight calculator module for finding the important keywords.

Stemmer*:* In natural languages, inflection between words exists. Several different words may have the same meaning. So, these words should be treated as one word. Therefore we use stemming in indexing. It is the process of transforming a word into their word stem or root form. For example ‘buses’ becomes ‘bus’ etc. It is used to reduce the size of the index. The most common algorithm for stemming is PORTER’S ALGORITHM [106]. It consists of 5 phases of word reduction. In all the phases rules are selected. Some rules of stemming are:

Rules	**Examples**
SSES→ SS	businesses → business
IES→I	authorities → authoriti
S →	dogs → dog
SS→SS	progress → progress

7.2.2 Weight Calculator (W): This module after getting the keywords, assigns weight to each depending upon various factors like frequency and the location of the keyword in its HTML. Weight W is calculated according to equation 7.1 below.

$$\boxed{\mathbf{W = (W_F + W_P + W_T) / 3} \qquad \text{(Eq. 7.1)}}$$

Where W_F is the weight assigned to the keyword calculated according to the frequency of word in the document. W_P is weight given to keyword according to its position in the document. If the keyword is present in HEAD or TITLE then it is assigned more weight, giving it more importance and when present in body then it is assigned less weight. As this work is proposing a new indexing technique for hidden web data, W_T is taken in account. This is the weight assigned when keyword is present in tags special to hidden web i.e. <*table*> and <*div*>.

Then according the average of values of total weights, the keywords having top ten values are taken and rest are discarded. These keywords are also stored temporarily in a set S_{url} corresponding to each page. This set S_{url} of keyword is enhanced with more information in next steps. The different weights are assigned such that the values can be normalised between 0 and 1.These values are chosen during experimentation after getting the hidden web pages by the Hidden Web Crawler. Detailed procedure to find the Weights of keywords is given below:

i) Frequency based Weights (W_F)*:* As the name suggest it calculates weight of keywords according to frequency of that word in particular document as shown in Table 7.1. for the web page www.ymcaust.in shown in figure 7.3(a). Input for this module is list of keywords generated by Keyword Extractor. Most frequently used word will have more weights and so on. The output of this module is a list of keywords having weights associated with them.

Table 7.1 Frequency Based Weights

Keyword	Frequency	W_F (Log(Frequency))
YMCA	21	1.32
Student	10	1
Department	19	1.27

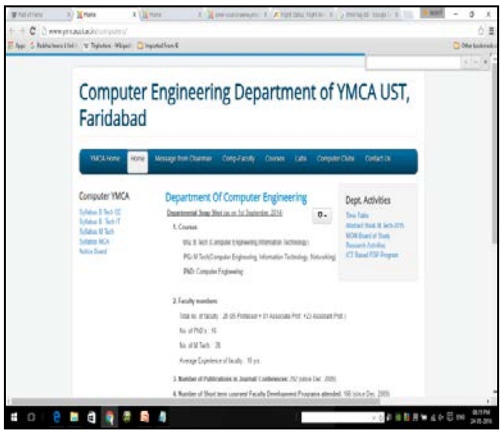
ii) Position based Weights (W_P)*:* Only Frequency based weights will not suffice as there may be a possibility that some spammer who wants their document at top search result can only write that word in the document. So, here the tag under which a keyword is placed in document is also given importance depicted in Table 7.2. For Example keywords which are enclosed under tags like <*Title*> and <*Head*> are given more weights as compared to the words present in <*body*> tag.

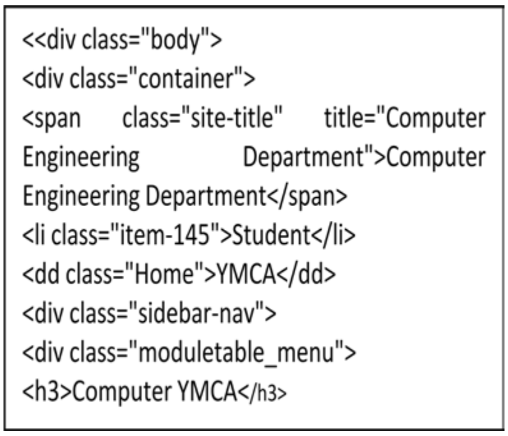
Table 7.2 Position based Weights

Keyword	Tag	Position based Weights(W_P)
Department	Title	0.5
YMCA	Heading	0.3
Student	Body	0.2

iii) Tag based Weights (W_T): In this work the focus is toward the indexing of hidden web data hence the tags present in hidden web pages are given importance and are considered in table based weights.

In figure 7.3a) and 7.3 b) a snapshot of hidden web page of YMCA University is shown along with the HTML code of the web page. Various Tags which contain the desired information is present in* the below specified tags. The most common tags among them are i) <*li*> (ii) <*div class*> iii) <*dt*><*dd*> tags which contains important information in between them.





```
<<div class="body">
<div class="container">
<span    class="site-title"    title="Computer
Engineering          Department">Computer
Engineering Department</span>
<li class="item-145">Student</li>
<dd class="Home">YMCA</dd>
<div class="sidebar-nav">
<div class="moduletable_menu">
<h3>Computer YMCA</h3>
```

Figure 7.3 a) Snapshot of Hidden Web Page of YMCA University Website

Figure 7.3 b) HTML Code for the Same.

As the data present in a table is of much importance in hidden web pages hence they are given higher weights. The keywords present between these tags are provided the highest value refer table 7.3.

Table 7.3 Tag based Weights

Keyword	Tag	Tag based Weights (W_T)
Department	<.li>	0.9
YMCA	<div class>	0.8
Student	<dd>	0.6

Based upon the values of W_F, W_P, W_T the final weight is calculated for the keywords by using equation 7.1 as shown in table 7.4. The final weight will decide which keywords will be taken to next phase for further analysis. The keywords which are having a weight greater than the threshold value are stored and taken for further processing in next module. The threshold value taken in this work is 0.8.

Table 7.4 Final Weight Table

Keyword	**Frequency based weight**	**Position based weight**	**Tag based weight**	**Average Weight**
YMCA	1.32	0.3	0.8	.80
Department	1.27	0.5	0.9	.60
Student	1	0.2	0.6	0.89

So in above case the keywords 'YMCA' and 'Student' are taken to next step and also a set of these keywords is made here.

$$\mathbf{S_{url} = \{YMCA, Student\}}$$

7.2.3 Concept Annotator

After getting top keywords from each web page they are matched with a predefined Ontology and the matched concepts (class) is stored with the keyword in the form of a table. These value plus class pairs are also stored in a set S'_{url} also. There may be words that are not directly matching with any class of ontology, then their synonyms are find out with the help of WORDNET and matched with ontology class equation 7.3 and 7.4.

S'_{url} = { keyword+ Class} If keyword exist in Ontology (Eq. 7.3.)

S'_{url} = {Synonym(Keyword)+Class} if not in Ontology (Eq. 7.4.)

For example for the values above YMCA is the keyword and the class it matches with is the University which is a child of education institute.

S'= {YMCA, UNIVERSITY}

This approach works well until the keyword chosen is not an ambiguous term but if we come across with an ambiguous term which matches to more than one classes in an ontology, are taken to the next step of context annotator. Few examples of such terms are: i) apple it is a fruit as well as a device name shown in figure 7.4 below ii) jaguar both a cat name a car name and a fighter plane name. To solve this problem the keywords are sent to next phase context Annotator. Example for concept annotation of ambiguous word: Suppose user enters apple now apple can be fruit and apple can be iphone. At this stage apple is mapped with both concepts device and fruit.

S'={apple, device, fruit }

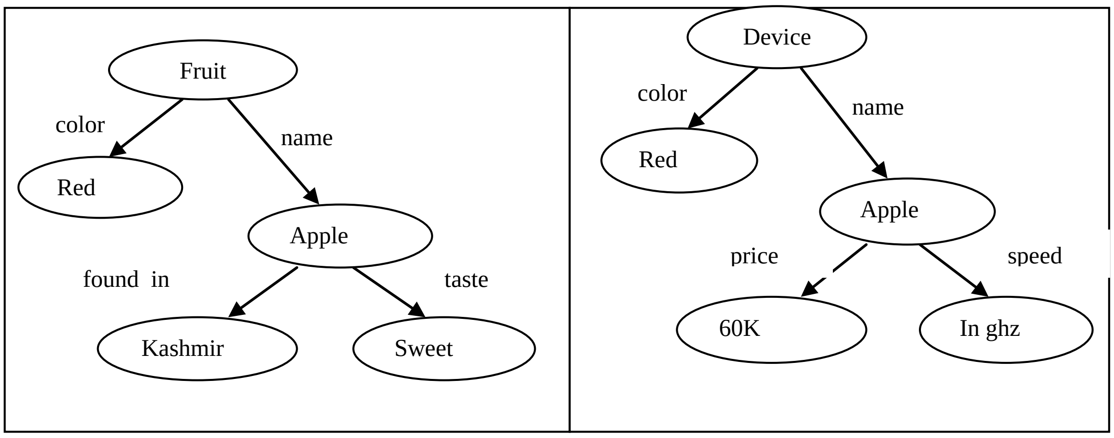


Figure 7.4 Example Showing Two Classes Matching for Ambiguous Term Apple.

7.2.4 Context Annotator

Context is any information attached with a word that is used to disambiguate or to find the situation where it is used. When the same word denotes two concepts (as in "red" color and "red" Communist), it is the context that differentiates between the two. It provides user with more relevant information.

Context of a keyword here is found by considering the superconcept(C), subconcept(C) and sibling(C) of a concept C and taking a union of each sub, super and concept itself as depicted in equation 7.5. Also the synonyms of concepts are taken with union operation.

Context(W) = Synonym(W) U Concept(W) U Parent(Concept(W)) U Sibling(Concept(W)) U Ancestor(Concept(W). (Eq. 7.5)

The synonyms can be found with the help of Wordnet[107] or thesaurus[108]. Example: If user inputs sweet or red apple then result should show the pages containing the information about fruit. If user enters fast apple then search system should provide user with apple iphones. Here this disambiguation is done by looking at the ontology graph where the parent node's information is stored as context of the keyword along with the keyword itself (table 7.5 for example).

Table 7.5 Final Index Table having Concept as Well as Context Associated with a Term

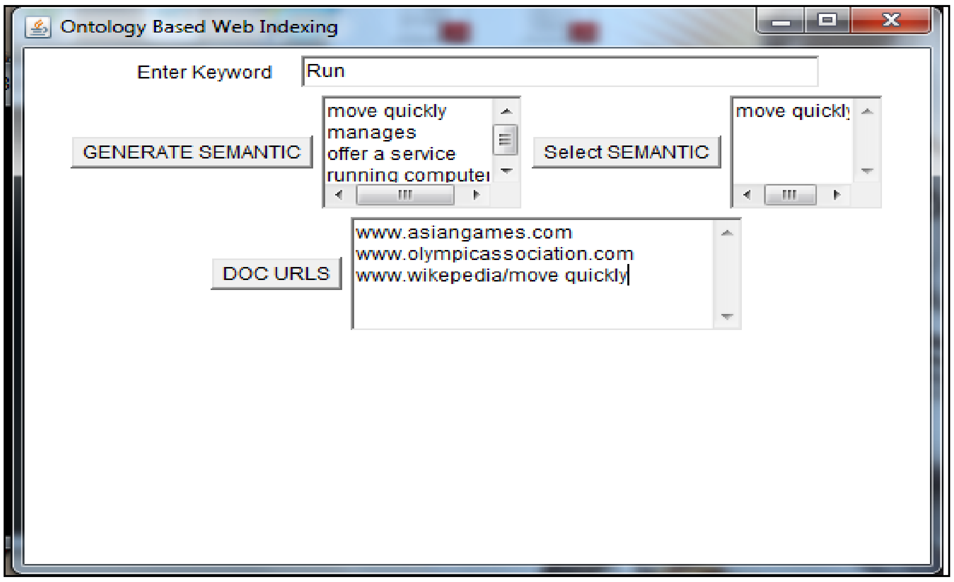
Keyword	**Concept(class)**	**DocId's**	**Context(Meaning)**
Mouse	Computer	1,4,6,7	Pointing Device
Mouse	Animal	2,3,5,8	Rodent
jaguar	Animal	1,3,4,7,11	Panther
jaguar	Fighter plane	2,6,8,10	Name
jaguar	Car	1,2,4	Automobile

7.2.5 IMPLEMENTATION OF ONTOLOGY BASED INDEXING SCHEME FOR HIDDEN WEB

A test set of web pages downloaded by hidden web crawler that contained synonyms and ambiguous words were selected and indexed. Figure 7.5 below shows the working of our experimental ontology based Web Indexing System.

1. First a Keyword is entered into the search interface of search engine.
2. Search Engine returns all contexts (semantics) attached with keyword.
3. User selects the desired context (semantic).
4. On the basis of context user choose, only the documents corresponding to the same context are chosen.

The indexing module indexes the documents provided and creates the index table conataining concept and context with each URL. The implementation screen shot shows that corresponding to one word there are more than one context



Run

move quickly
manages
offer a service
running computer

move quickly

www.asiangames.com
www.olympicassociation.com
www.wikepedia/move quickly

Figure 7.5. Snapshot of Experimental Setup of Indexing System.

Depending upon the context the inbdexer had already classified the URL’s.When a user selects a particular context only those URLs are displayed which have the similar context.The implementation shows that the indexing tables are correctly made at the backened.

7.3 RANKING MODULE

Page ranking is the process of arranging the URL’s of pages in an ordered manner to present them to user. The ordering can be in order of relevance and importance of the web pages. Without ranking the user has to traverse many pages in order to get desired result. Researchers have given many ideas to improve the ranking but most of the ideas are given for surface web. Very little work has been done for the high quality hidden web pages. Also some of the works done so far are using query dependent factors for ranking and some uses query independent factors. None has tried to use both factors for improving the relevancy.

The algorithm developed in this work is considering the hidden web data as a factor for ranking. The architecture of the proposed ranking technique is shown in figure 7.6. This algorithm uses factors for both query dependent and query independent ranking methods. Query dependent factors such as page frequency , query – page content matching are used and query independent factors such as page content popularity , user feedback are also used to develop a novel architecture.

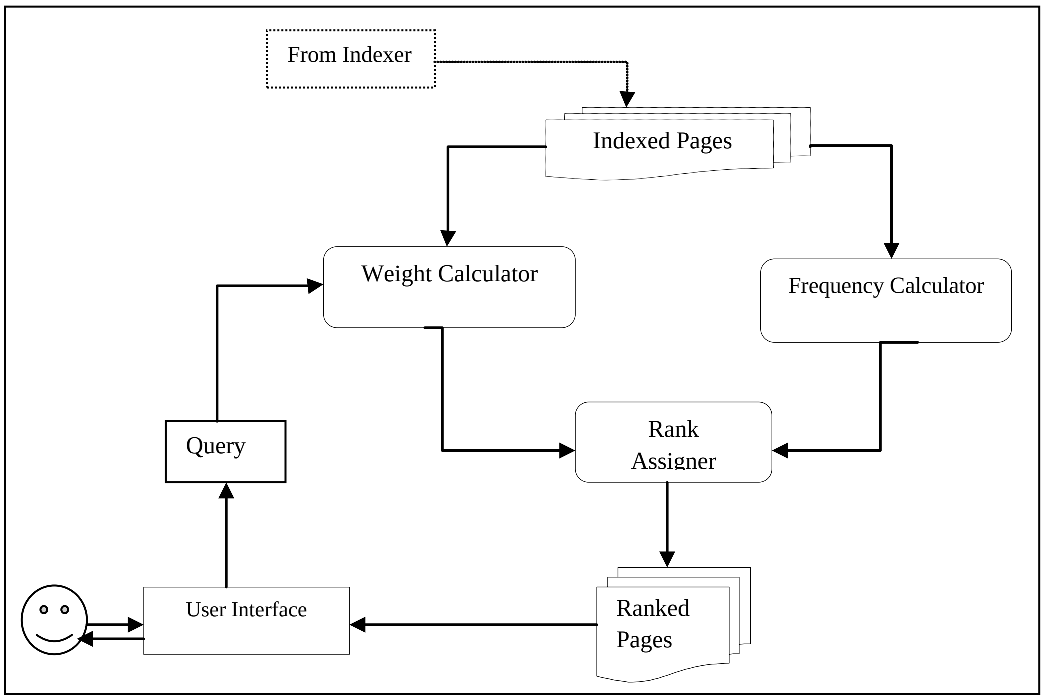


Figure 7.6 Architecture of Proposed Ranking Technique for Hidden Web Pages

The major components of proposed Architecture are as follows:

i) Weight Calculator
ii) Frequency Calculator
iii) Rank Assigner

The details of each component are discussed below:

7.3.1 Weight Calculator (W): In this module weight W is assigned to each url/web page on basis of three factors. One is rating on the web page (W_1), second Users Feedback present on the web page (W_2) and third is Query-Page Content Matching (W_3), as shown in figure 7.7 below. These factors are important in hidden web for example in book domain the rating of a book and user’s feedback for a book plays important role in finding an important book. The URL having good rating should come above in the list and same is true for user’s feedback. The detailed explanation on how to calculate the same is given in below subsection.

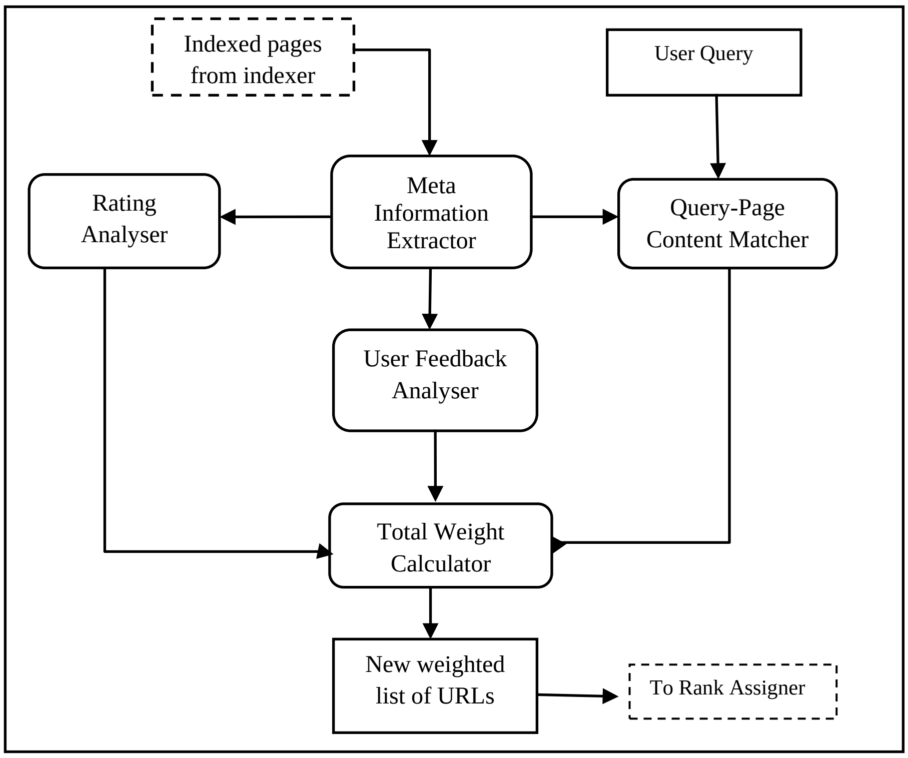

Indexed pages from indexer
User Query
Rating Analyser
Meta Information Extractor
Query-Page Content Matcher
User Feedback Analyser
Total Weight Calculator
New weighted list of URLs
To Rank Assigner

Figure 7.7 Components of Weight Calculator Module

i) Meta Information Extractor: This subcomponent takes the URL from index table. Match the URL to the documents present in the repository and extract all the information attached with the document. This information is used by other subcomponents to calculate weight and other factors required for ranking.

ii) Rating Analyser (W_1): Rating on any web page shows the popularity of the website owner or service provider on web. There can be ratings given to individual book also whether the book is worth reading or not. In case of e-commerce websites ratings are given by users/buyers for any product they had already purchased. This rating is helpful in calculating the relevancy of the desired result retrieved by submitting the query.

Otherwise rating on book web page may be a bookseller’s/publisher’s rating which is based on seller’s completion rate. Completion rate represents a seller's percentage of successfully completed orders that is the number of orders a seller receives versus the number of orders cancelled or returned. Booksellers with a higher Bookseller Rating have cancelled fewer orders and received fewer returns. Ratings on various websites are normally shown in form of stars. Meaning of these star rating is as follows.

5 full stars :- W_1 is assigned maximum weight taken 5 here.

4 full stars :- W_1=4.

3 full stars :- W_1=3 and so on.

Thus rating analyser will extract this rating present on the web page and assign these values.

iii) **Users Feedback Analyser (W_2):** On web pages there are some sort of user's feedbacks available also, which users have provided in form of likes/ dislikes, user reviews etc. These feedback shows the usability of the page content means whether the information/product available on the webpage is up to the requirement or not, whether user liked it or not. User Feedback analyzer will extract these reviews from the web page and try to analyze them and set W_2 which is a combination of two weights W_{21} an W_{22} a numerical value as shown

$$W_2 = W_{21} + W_{22}$$

i) if no. of likes>no. of dislikes then set $W_{21}=1$

if no. of dislikes>no. of likes then set $W_{21}=0$.

ii) set $W_{22}=1$ for +ve reviews otherwise set $W_{22}=0$.

For analysing positive reviews or likes on a page few words like good, Excellent and worth are considered. Similarly for taking dislikes counts the negative words like bad, weak, poor and not worth are taken into consideration.

iv) **Query-Page Content Matching (W_3):** This is simple keyword matching scheme as traditionally applied to calculate rank according to the query. In this sub module the query entered by user is matched with the content of the web page. The similarity percentage between query and web page's indexed information is calculated. Such as web page having a book shows its title as 'computer programming' and author as 'Y. S. Singh' and user enters a query having book title 'computer' and author 'Singh' then the matching % is calculated between user query and title and author on the webpage.

This matching is done by using the Levenshtein distance (edit distance***)*** method []. The Levenshtein distance is a string metric for measuring the difference between two sequences. Informally, the Levenshtein distance between two words is the minimum number of single-character edits (i.e. insertions, deletions or substitutions) required to change one word into the other. Percentage matching is calculated by following formula:

$$\%\ \text{matching} = \frac{\text{Total characters in query} - \text{Number of character edits}}{\text{Total number of characters in query}} \qquad \text{(Eq. 7.6)}$$

v) Total Weight Calculator (W): Weight W can be calculated as a total of W_1,W_2 and W_3 as shown by algorithm below in figure 7.8

```
Step 1: Take New URL Lists and extract their meta information such as Title ,
        Authors,   Rating, Like/dislike of Product or Webpage content .
     2: Set values of sub-weight $W_1$ ,$W_2$,$W_3$  as following -
        a) Check rating of web page , Set weight w1 according to rating as-
           $W_1=5$ for 5 star rating $W_1= 4$ for 4 star rating and so on.
        b) Check for like/dislikes or reviews given by user as-
           if likes and dislikes are given then set $W_2$ as-
              if no. of likes>no. of dislikes then set $W_{21}=1$
              if no. of dislikes>no. of likes then set $W_{21}=0$
           If reviews given then try to analyze some reviews and if reviews are
           positive then set $W_{22}=1$ otherwise $W_{22}=0$
                          Set $W_2=W_{21}+W_{22}$
     3: Match the respective content of web page with query entered by user. If
        80% matches then set $W_3=80/10$ and so on.
     4: Calculate total weight as $W=W_1+W_2+W_3$
     5: Return the url_lists having weights assigned to each url.
```

Figure 7.8 Algorithm for Total Weight Calculator Module

7.3.2 Frequency Calculation Module: As the hidden web crawler hidden web pages by filling HTML form on the various websites. It is possible that hidden web crawler fill same set of values in HTML form on more than one website. It may result in generation of web pages having same content from two different websites. Such as two websites of book domain may generate two web pages with different url having description of same book. Frequency calculator will calculate such number of pages and set this as frequency of web page. This frequency is taken in to account to calculate rank as the pages having higher frequency are given preferences hence given higher ranks. Other URL's will be assigned less ranks as shown in figure 7.9 below.

```
Step 1: Arrange the weighted url lists in decreasing order of weight.
     2: Set frequency of every url as 1.
     3: Take urls having similar weight, check for the content of a url
        and match it with others having similar weight, if content/book
        is same then increase the frequency, f by the number of urls
        having same result.
     4 : Return a url list having frequency  attached with them..
```

Figure 7.9 Algorithm for Frequency Calculator Module

Here the frequency of web pages having similar result is higher as compared to the URLS having different results. This factor is especially taken in to consideration for hidden web.

7.3.3 Rank Assigner: In this module final rank value is assigned to each URL. Rank value is used to arrange all URL's in a ranked list. This rank value, Rv is calculated on basis of weight assigned to each URL(W), and frequency of the web page(f) as:

$$Rv = W * f$$

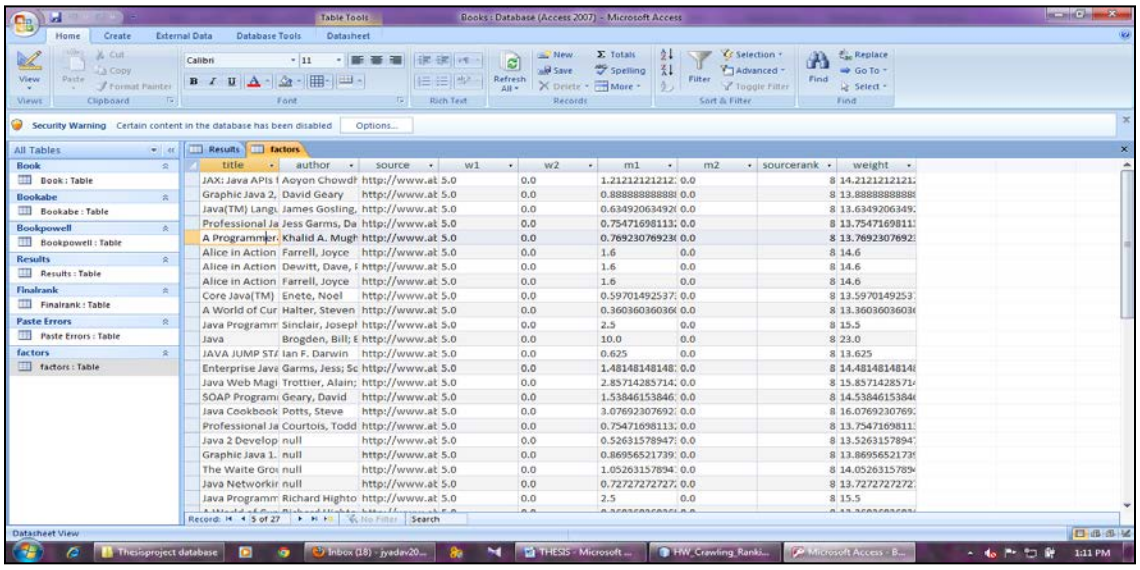
```
Step 1: Take Url list , each url having Weight (W) and Frequency(f).
     2: Calculate Rank Value (Rv) for each url as-
                $Rv=(W*f)$
     3: Arrange All Urls in descending order of Rank value (Rv). Url having Highest
        value of Rv will be ranked higher.
     4: Store & Display the Ranked Urls to user.
```

Figure 7.10 Algorithms for Final Rank Assigner

After calculating rank value of each retrieved url , store the rank value and display the results to the user after arranging the rank value in descending order which means the url or web page having highest rank value will get maximum priority and will be ranked at first position.

7.4 EXPERIMENTAL RESULTS

The proposed ranking technique is implemented for book domain. Below are some important screenshots showing illustration of the proposed technique. Hidden web index repository has already been created to store all the crawled or downloaded pages by hidden web crawler. The indexed data is shown in figure 7.11. It is showing all the information of web pages such as Author name, Title, Rating, feedback related to a keyword. The keyword here is java which is being entered by user at the interface provided.

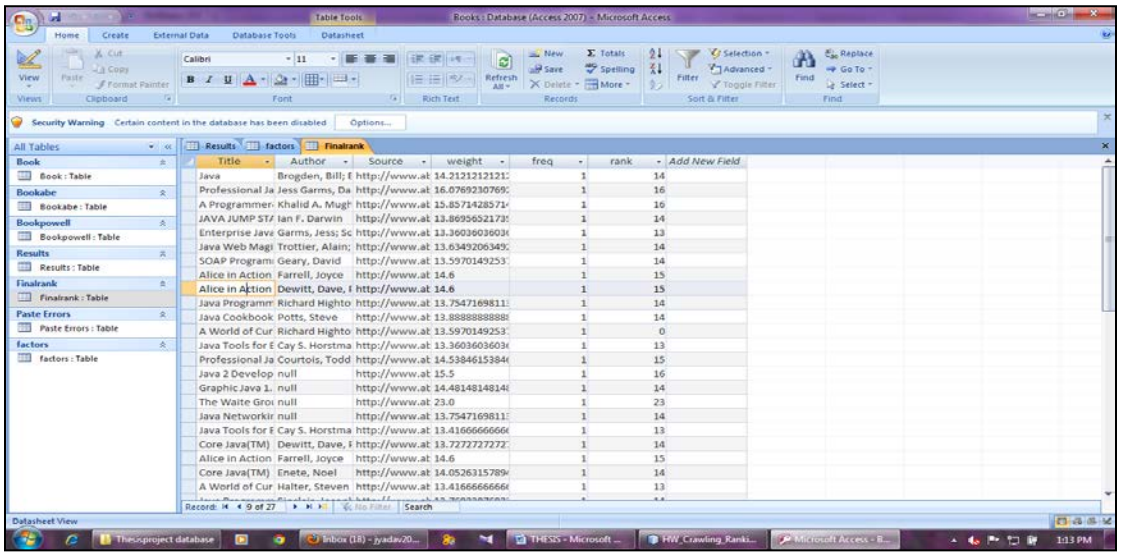


title	author	source	w1	w2	m1	m2	sourcerank	weight
JAX: Java APIs f	Aoyon Chowdh	http://www.at	5.0	0.0	1.2121212121	0.0	8	14.2121212121
Graphic Java 2,	David Geary	http://www.at	5.0	0.0	0.8888888888	0.0	8	13.8888888888
Java(TM) Langu	James Gosling,	http://www.at	5.0	0.0	0.6349206349	0.0	8	13.6349206349
Professional Ja	Jess Garms, Da	http://www.at	5.0	0.0	0.7547169811	0.0	8	13.7547169811
A Programmer.	Khalid A. Mugh	http://www.at	5.0	0.0	0.7692307692	0.0	8	13.7692307692
Alice in Action	Farrell, Joyce	http://www.at	5.0	0.0	1.6	0.0	8	14.6
Alice in Action	Dewitt, Dave, F	http://www.at	5.0	0.0	1.6	0.0	8	14.6
Alice in Action	Farrell, Joyce	http://www.at	5.0	0.0	1.6	0.0	8	14.6
Core Java(TM)	Enete, Noel	http://www.at	5.0	0.0	0.5970149253	0.0	8	13.5970149253
A World of Cur	Halter, Steven	http://www.at	5.0	0.0	0.3603603603	0.0	8	13.3603603603
Java Programm	Sinclair, Joseph	http://www.at	5.0	0.0	2.5	0.0	8	15.5
Java	Brogden, Bill; E	http://www.at	5.0	0.0	10.0	0.0	8	23.0
JAVA JUMP STA	Ian F. Darwin	http://www.at	5.0	0.0	0.625	0.0	8	13.625
Enterprise Java	Garms, Jess; Sc	http://www.at	5.0	0.0	1.4814814814	0.0	8	14.4814814814
Java Web Magi	Trottier, Alain;	http://www.at	5.0	0.0	2.8571428571	0.0	8	15.8571428571
SOAP Program	Geary, David	http://www.at	5.0	0.0	1.5384615384	0.0	8	14.5384615384
Java Cookbook	Potts, Steve	http://www.at	5.0	0.0	3.0769230769	0.0	8	16.0769230769
Professional Ja	Courtois, Todd	http://www.at	5.0	0.0	0.7547169811	0.0	8	13.7547169811
Java 2 Develop	null	http://www.at	5.0	0.0	0.5263157894	0.0	8	13.5263157894
Graphic Java 1.	null	http://www.at	5.0	0.0	0.8695652173	0.0	8	13.8695652173
The Waite Grou	null	http://www.at	5.0	0.0	1.0526315789	0.0	8	14.0526315789
Java Networkir	null	http://www.at	5.0	0.0	0.7272727272	0.0	8	13.7272727272
Java Programm	Richard Highto	http://www.at	5.0	0.0	2.5	0.0	8	15.5

Record: 5 of 27

Figure 7.11 Indexed Data with All the Calculated Intermediate Weight Values

Along with the above said information the weight calculated according to equation 7.7 and frequency of the page according to similar content is also stored**.** The total weight W is calculated as total of sub weights i.e ranking factor, feedback factor and query content matching factors. Now Frequency of the pages is calculated and stored as shown in figure 7.12.

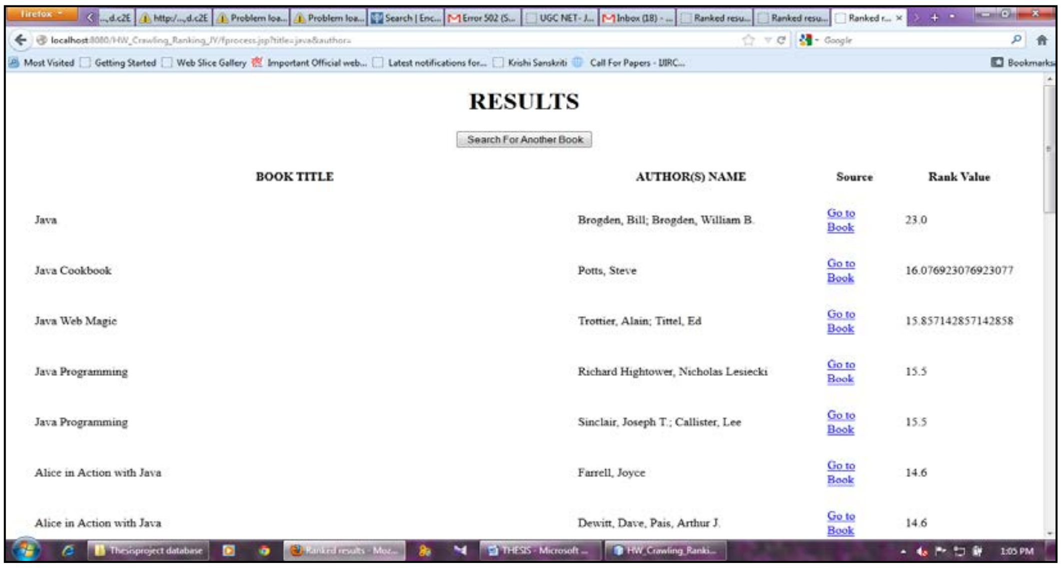


Title	Author	Source	weight	freq	rank	Add New Field
Java	Brogden, Bill; E	http://www.at	14.2121212121	1	14	
Professional Ja	Jess Garms, Da	http://www.at	16.0769230769	1	16	
A Programmer	Khalid A. Mugh	http://www.at	15.8571428571	1	16	
JAVA JUMP STA	Ian F. Darwin	http://www.at	13.8695652173	1	14	
Enterprise Java	Garms, Jess; Sc	http://www.at	13.3603603603	1	13	
Java Web Magi	Trottier, Alain;	http://www.at	13.6349206349	1	14	
SOAP Program	Geary, David	http://www.at	13.5970149253	1	14	
Alice in Action	Farrell, Joyce	http://www.at	14.6	1	15	
Alice in Action	Dewitt, Dave, F	http://www.at	14.6	1	15	
Java Programm	Richard Highto	http://www.at	13.7547169811	1	14	
Java Cookbook	Potts, Steve	http://www.at	13.8888888888	1	14	
A World of Cur	Richard Highto	http://www.at	13.5970149253	1	0	
Java Tools for E	Cay S. Horstma	http://www.at	13.3603603603	1	13	
Professional Ja	Courtois, Todd	http://www.at	14.5384615384	1	15	
Java 2 Develop	null	http://www.at	15.5	1	16	
Graphic Java 1.	null	http://www.at	14.4814814814	1	14	
The Waite Grou	null	http://www.at	23.0	1	23	
Java Networkir	null	http://www.at	13.7547169811	1	14	
Java Tools for E	Cay S. Horstma	http://www.at	13.4166666666	1	13	
Core Java(TM)	Dewitt, Dave, F	http://www.at	13.7272727272	1	14	
Alice in Action	Farrell, Joyce	http://www.at	14.6	1	15	
Core Java(TM)	Enete, Noel	http://www.at	14.0526315789	1	14	
A World of Cur	Halter, Steven	http://www.at	13.4166666666	1	13	

Record: 9 of 27

Figure 7.12 Frequency and Rank Value.

Rank Value Calculation: After Frequency calculation, using all the factors rank value is assigned to each page. Now urls be ranked in descending order means higher the value higher will be priority of the page. Figure 7.13 shows the final rank along with each URL which is presented to user.

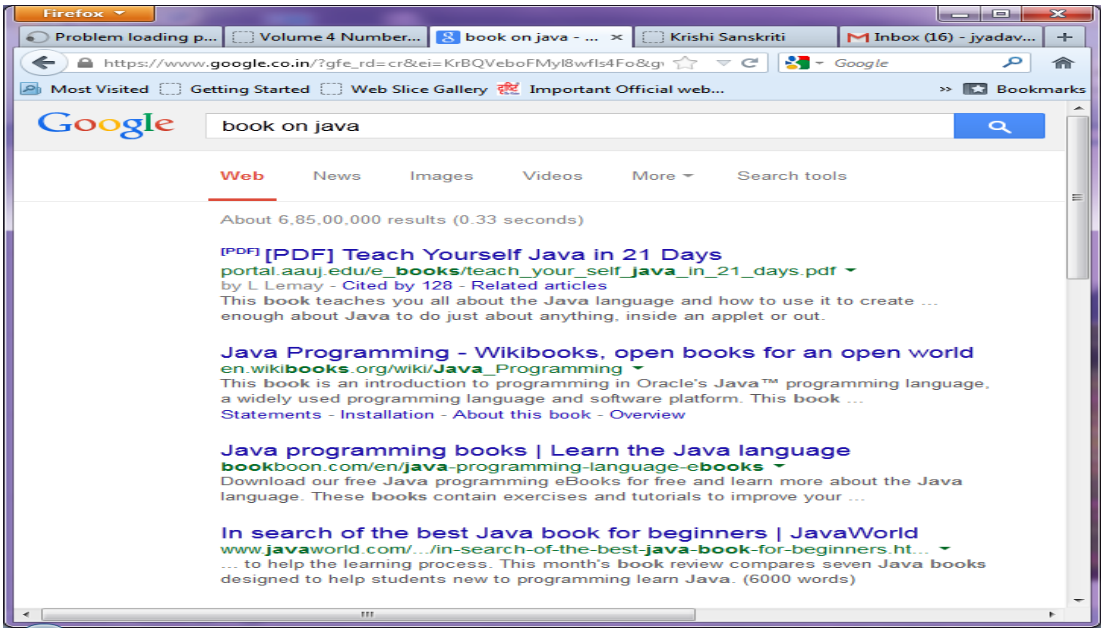


RESULTS

BOOK TITLE	AUTHOR(S) NAME	Source	Rank Value
Java	Brogden, Bill; Brogden, William B.	Go to Book	23.0
Java Cookbook	Potts, Steve	Go to Book	16.076923076923077
Java Web Magic	Trottier, Alain; Tittel, Ed	Go to Book	15.857142857142858
Java Programming	Richard Hightower, Nicholas Lesiecki	Go to Book	15.5
Java Programming	Sinclair, Joseph T.; Callister, Lee	Go to Book	15.5
Alice in Action with Java	Farrell, Joyce	Go to Book	14.6
Alice in Action with Java	Dewitt, Dave, Pais, Arthur J.	Go to Book	14.6

Figure 7.13 Ranked Results to User Along with the Rank Value of Each URL.

The results of proposed ranking technique are compared with general ranking technique used for general web.



book on java

About 6,85,00,000 results (0.33 seconds)

[PDF] Teach Yourself Java in 21 Days
portal.aauj.edu/e_books/teach_your_self_java_in_21_days.pdf
by L Lemay - Cited by 128 - Related articles
This book teaches you all about the Java language and how to use it to create ... enough about Java to do just about anything, inside an applet or out.

Java Programming - Wikibooks, open books for an open world
en.wikibooks.org/wiki/Java_Programming
This book is an introduction to programming in Oracle's Java™ programming language, a widely used programming language and software platform. This book ...
Statements - Installation - About this book - Overview

Java programming books | Learn the Java language
bookboon.com/en/java-programming-language-ebooks
Download our free Java programming eBooks for free and learn more about the Java language. These books contain exercises and tutorials to improve your ...

In search of the best Java book for beginners | JavaWorld
www.javaworld.com/.../in-search-of-the-best-java-book-for-beginners.ht...
... to help the learning process. This month's book review compares seven Java books designed to help students new to programming learn Java. (6000 words)

Figure 7.14 Result on Google for The Same Query Book on Java.

It can be easily observed by figure 7.14 that Google doesn’t provide the book name as such as a result. It is showing the links of various websites further where user has to visit and fill all the necessary fields then only will get the desired result. Whereas in proposed system the user is getting the name, author, URL and all the other information on the first page. The URL can be clicked and the system will take the user to the corresponding book improving the relevancy and user satisfaction. The next chapter discusses the conclusion and future work of the system.

Chapter 8

CONCLUSION AND FUTURE WORK

8.1 CONCLUSION

In this dissertation, ontology based information retrieval system for retrieving hidden web data has been developed. The rich and efficient ontology has been developed using a novel approach. A new ontology based hidden web crawler has been deployed to collect the hidden web data efficiently. More specifically, the main challenges involved in developing a crawler for the Hidden Web have been addressed and resolved. For making the crawler work properly an effective ontology mapping technique has been devised. This technique improves the efficiency and reduces the resources (time and network bandwidth) required. The proposal and implementation of novel indexing and ranking technique has resolved the issue of ambiguity and improved the relevancy of the system. The whole system is developed which also resolve the issue of synchronization.

The proposed work meets the following objectives:

- **Relevance:** Ontology involves the context of the data and relationship between terms, which improves the process of finding the exact values to be filled on form interfaces. Getting the appropriate values to be filled obtain good results with high precision, thereby improving the relevance of the system.
- **Database Selection and Implementation:** A domain specific semantic database has been selected and created for finding appropriate values which are required to be filled in form interface. This semantic database is stored in Oracle 10g in form of Subject, Predicate and Object triples. These records in form of <S,P,O> are satisfy users queries more suitably.
- **Automatic and Rich Ontology**: A new technique for automatic generation of ontology with the help of web pages is developed. This approach attaches meaning to the data for extracting data hidden behind form interfaces. A scalable approach to gather hidden web data by filling forms automatically has been designed and develop

- **Efficiency in Mapping:** A new effective mapping technique has been proposed which improves efficiency of the hidden web crawler by removing pairing of irrelevant nodes during ontology mapping. Ontology Mapping System developed here also considering the proper utilization of resources like time and space.
- **Synchronization:** The integration of various phases for developing a whole information retrieval system for hidden web has been designed and implemented. All the processes like form filling, submitting, downloading the web pages, indexing them and presenting to user has been done in various phases.
- **Ambiguity in Indexing*:*** A novel ontology based indexing scheme has been designed which not only attaches the concept with terms, but also removes ambiguities between terms using the context for the same. A well formed index table constructed as a result of indexing proves that the indexing is well suited for hidden web data.
- **Ranking**: A new ranking technique that involves both query dependent and query independent factors for hidden web has been proposed and implemented. This technique improves the precision and recall of the overall system.

The performance of hidden web crawler has improved significantly as Precision, Recall and F-measure are found to be improved than the traditional hidden web Crawler.

Comparing with previous DSHWC [40] there is 3.9% improvement in Precision, 2.9% improvement in Recall and 8.8% of improvement in f-measures for book domain. Similarly there is an improvement of 9.5% increase in Precision, 6.3% in Recall and 7.6% of improvement in f-measures for airline domain.

8.2 SCOPE FOR FUTURE WORK: Some of the possible issues that could be further explored or extended in future are as follows:

i) **Query Prediction**: In this work Hidden web Crawler is generating queries so as to download the required page. To reduce the load for the crawler a query prediction technique can be proposed and applied so as to find which query should be fired in the next round.

ii) **Learning/Training of the system:** The whole system can be made more efficient if it will apply some training schemes. These schemes can be applied at the crawler part during matching process so that the high precision can be gained while finding values to be filled at form interface. Also during query formation the training data can be used to predict the next query to be fired reducing network bottleneck.

APPENDIX

IMPLEMENTATION SNAPSHOTS OF OVERALL SYSTEM

Implementation of the whole system is done using Netbeans 8.0 IDE and Microsoft SQL server. The already developed ontology is saved as semantic database in Oracle10g. The database constructed is in the form of Subject, Object and Predicate triples. Given below are few snapshots of developed Ontology based Information Retrieval System for Hidden Web.

1. **Startup Screen:** Figure 1 below shows the startup screen where two options to work further are given.

i) First by clicking on Admin we start the backend processes of ontology creation, hidden web crawling and indexing. And by clicking on User an interface interface is provided to user to enter query and the results are shown as per query entered

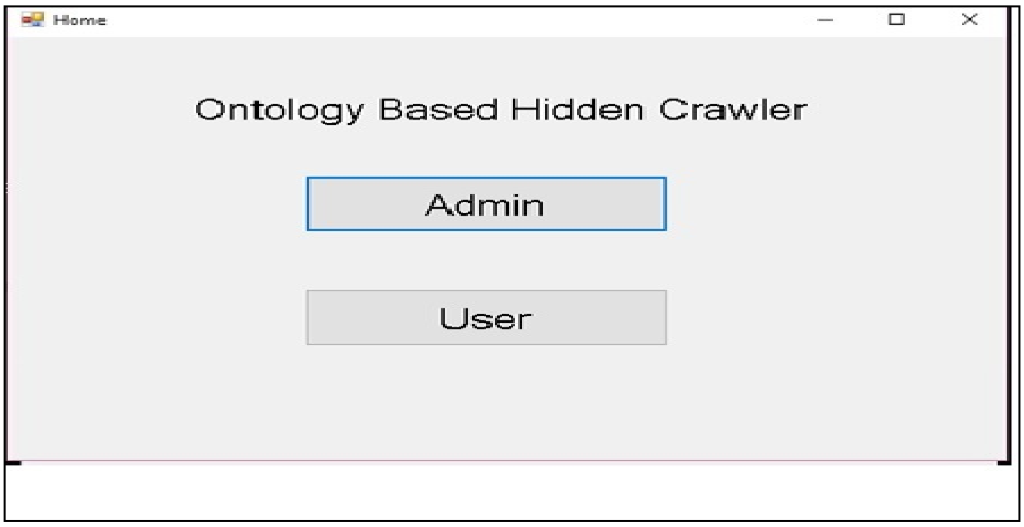


Figure 1 The startup screen of Ontology based Information Retrieval System

ii) The next screen by clicking on admin comes where there are again two buttons one for ontology master and second for DB creation. The ontology master module contains the ontology created by Protégé in chapter 4. This ontology is merged with the ontology created by DB creation button. On clicking the ontology master the screen shows the graph of ontology created manually having various instances.

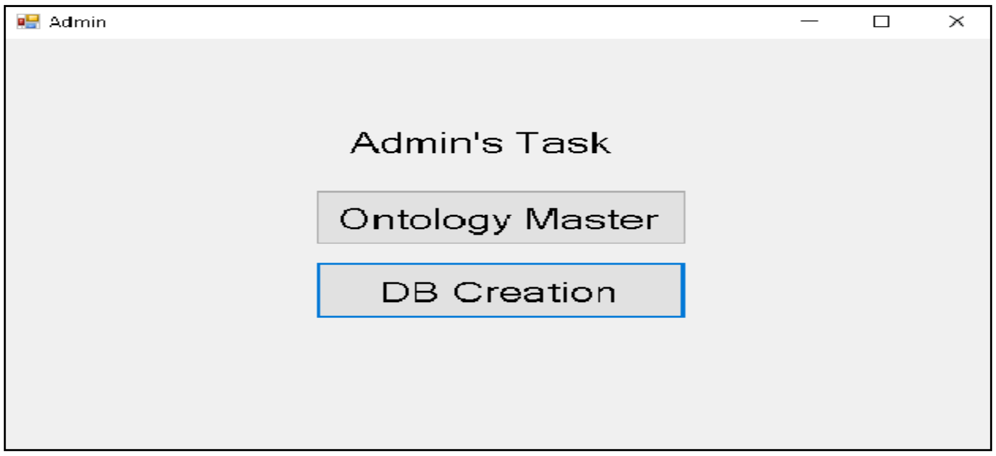
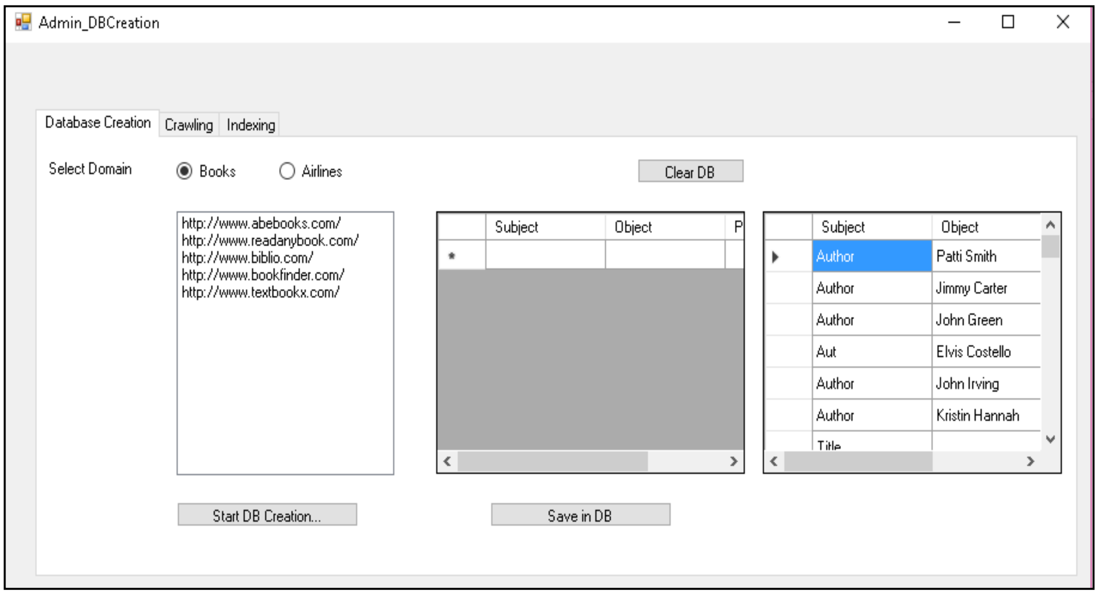


Figure 2 Admin Screen for two backend processes

2. Database Creation: Figure below shows that the system is taking two major domains of pages for generating the ontology i) Book ii) Airline. This screen shows the 5 interfaces of book domain taken upto extract the information in form of subject, object, predicate i.e. <S,P,O>. The snapshot shows various book domain websites. By clicking of any website the data from that particular site is extracted and stored in the table of <S,P,O> triples.



Select Domain [✓ Books] Airlines

http://www.abebooks.com/
http://www.readanybook.com/
http://www.biblio.com/
http://www.bookfinder.com/
http://www.textbookx.com/

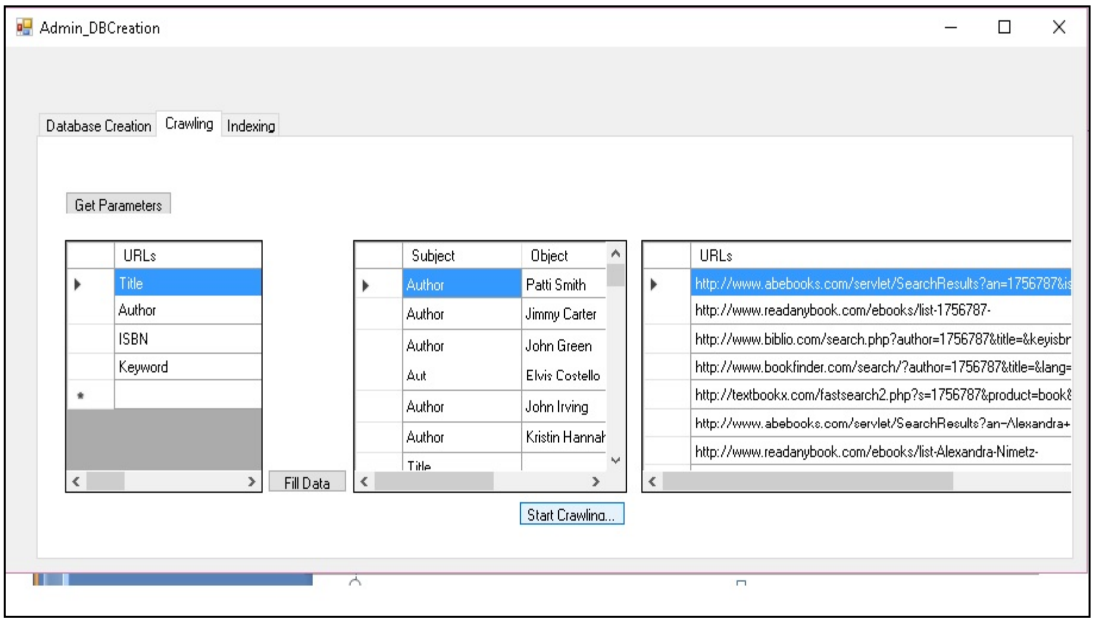
Subject	Object	P

Subject	Object
Author	Patti Smith
Author	Jimmy Carter
Author	John Green
Aut	Elvis Costello
Author	John Irving
Author	Kristin Hannah
Title	

Figure 3 Form Interfaces of Book Domain and Extracting Information from Them

3. The Crawling process of the system

Figure 4 shows crawling process of the system where the information stored above is used to generate queries and those queries are fired on WWW to get the desired hidden web data which is further and indexed and also presented to user according to user’s query. The data is taken from <S,P,O> table and the queries are constructed from the same using the format of the corresponding website where query is to be fired.



URLs
Title
Author
ISBN
Keyword

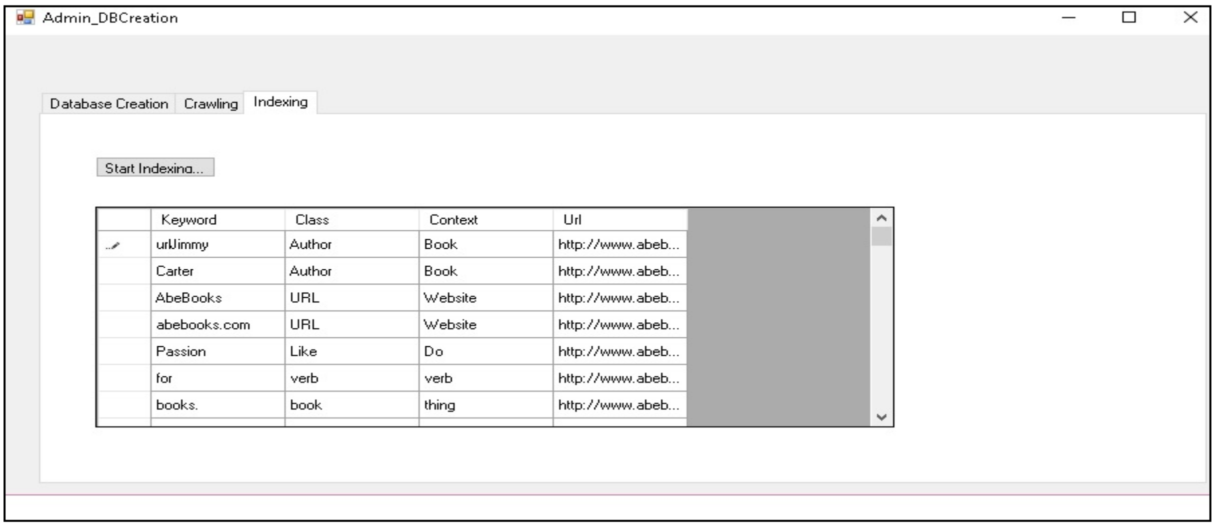
Subject	Object
Author	Patti Smith
Author	Jimmy Carter
Author	John Green
Aut	Elvis Costello
Author	John Irving
Author	Kristin Hannah
Title	

URLs
http://www.abebooks.com/servlet/SearchResults?an=1756787&is
http://www.readanybook.com/ebooks/list-1756787-
http://www.biblio.com/search.php?author=1756787&title=&keyisbr
http://www.bookfinder.com/search/?author=1756787&title=&lang=
http://textbookx.com/fastsearch2.php?s=1756787&product=book&
http://www.abebooks.com/servlet/SearchResults?an=Alexandra+
http://www.readanybook.com/ebooks/list-Alexandra-Nimetz-

Figure 4 The Crawling Process of OHWC

4. Indexing of the pages Retrieved by Hidden Web Crawler

A local repository of the web pages retrieved has been made. Keywords from the pages are extracted and indexing technique as mentioned in proposed work has been applied to find the class and context for each important keyword.



Keyword	Class	Context	Url
urlJimmy	Author	Book	http://www.abeb...
Carter	Author	Book	http://www.abeb...
AbeBooks	URL	Website	http://www.abeb...
abebooks.com	URL	Website	http://www.abeb...
Passion	Like	Do	http://www.abeb...
for	verb	verb	http://www.abeb...
books.	book	thing	http://www.abeb...

Figure 4 The Indexing Process of OHWC

REFRENCES

[1] "Web Crawling and Indexing", The Stanford natural language and processing, http://nlp.stanford.edu/IR-book/pdf/20crawl.pdf.

[2] Google, "*Google search engine*" http://www.google.com.

[3] Sergey Brin and Lawrence Page, "*The anatomy web search engine*", Proc. of 7th International World Wide Web Conference, volume 30, Computer Networks and ISDN Systems, pp 107-117, April 1998

[4] Ricardo Baeza-Yates and Berthier Ribeirmobileo-Neto, "Modem Information Retrieval", ACM Press/ Addison-Wesley, 1999.

[5] Arvind Arasu, Junghoo Cho, Hector Garcia-Molina, Andreas Paepcke, and Sriram Raghavan, "Searching the Web", ACM Transactions on Internet Technology (TOIT), l(l):2-43, August 2001 Terrence

[6] A. Brooks, "Web Search: How the Web has changed information retrieval", Information Research, April 2003.

[7] Peter Haase, Nenad Stojanovic, York Sure, and Johanna Volker, "Personalized Information Retrieval in Bibster,a Semantics-Based Bibliographic Peer-to-Peer System

[8] M.K. Bergman, "The Deep Web: Surfacing Hidden Value", September 2001, http://www.brightplanet.com/deepcontent/tutorials/DeepWeb/deepwebwhitepaper.pdf

[9] Bin He, Mitesh Patel, Zhen Zhang, Kevin Chen-Chuan Chang, "Accessing the Deep Web: A Survey", Communications of ACM version: 50, April 2004.

[10] Chris Sherman and Garyprice : "The invisible web: uncovering sources search engines can't see "

[11] B. Chandrasekaran and John R. Josephson," What Are Ontologies, and Why Do We Need Them?", Owl web ontology language reference. http://www.w3.org/TR/owl-ref/.

[12] Alexandro Ntoulas, "Downloading Textual Hidden-Web Content through Keyword Queries", University of California Los Angeles, Computer Science Department, In Proceedings of the Joint Conference on Digital Libraries (JCDL), 2005, Denver, USA.

[13] A. K. Sharma, Komal Kumar Bhatia, "A Framework for Domain-Specific Interface Mapper (DSIM)", International Journal of Computer Science and Network Security, VOL.8 No.12, December 2008.

[14] J. Cope, N. Craswell, and D. Hawking, "Automated Discovery of Search Interfaces on the Web", In Proc. of ADC, pages 181-189, 2003.

[15] Ian Horrocks, "Ontologies and the Semantic Web", Communication ACM 2008;51:58–67.

[16] Mike Burner, "Crawling towards Eternity: Building an archive of the World Wide Web" Web TechniquesMagazine, 2[5], May 1997.

[17] Luciano Barbosa, Juliana Freire, "An Adaptive Crawler for Locating Hidden Web Entry Points", IW3C2 2007, May 8–12, 2007, Banff, Alberta, Canada.

[18] A. K. Sharma, Komal Kumar Bhatia, "Automated Discovery of Task Oriented Search Interfaces through Augmented Hypertext Documents", Proc. First International Conference on Web Engineering & Application (ICWA2006).

[19] S.Raghavan and H. Garcia-Molina, "Crawling the hidden web", in VLDB, 2001, Stanford Digital Libraries Technical Report. Retrieved 2008-12-27.

[20] J. Lage, A. Silva, P. Golgher, and A. Laender. "Automatic generation of agents for collecting hidden web pages for data extraction", Data & Knowledge Engineering Volume 49, Issue 2 (May 2004).

[21] Ricardo Baeza-Yates and Carlos Castillo, "Crawling the infinite Web: five levels are enough", In Proceedings of the third Workshop on Web Graphs (WAW), volume 3243 of Lecture Notes in Computer Science, pages 156-167, Rome, Italy, October 2004. Springer

[22] Sunny Lam, "The Overview of Web Search Engines", 2009-04-15, ai: CiteSeerX.psu:10.1.1.28.6344.

[23] S. Brin and L. Page. "The Anatomy of a Large-Scale Hypertextual Web Search Engine", Proc. 7th WWW Conference, 1998.

[24] Andrew S. Tanenebaum, "Distributed Systems: Principles & Paradigms", www.cs.vu.nl/~ast/books/ds1/11.pdf

[25] McPherson, Stephanie Sammartino, "Tim Berners-Lee: Inventor of the World Wide Web", Twenty-First Century Books, 2009.

[26] J. Cho and A. Ntoulas. "Effective Change Detection Using Sampling." In Proceedings of the International Conference on Very Large Databases (VLDB), **2002.**

[27] Anuradha and A.K.Sharma : "A Novel Technique for Data Extraction from Hidden Web Databases" in International Journal of Computer Applications, 2011.

[28] Ricardo Baeza-Yates and Berthier Ribeirmobileo-Neto, "Modem Information Retrieval", ACM Press/ Addison-Wesley, 1999.IR

[29] Christopher D.Mannig , Prabhakar Raghavan and Hinrich Schutze, “Chapter 8: Evaluation in information retrieval”, 2009.

[30] S. Chakrabarti, K. Punera, and M. Subramanyam, “Accelerated focused crawling through online relevance feedback”, In Proc. of WWW, pages 148-159, 2002

[31] M. Gray. “The World-Wide Web Wanderer”, Available from URL: http://www.mit.edu:8001/people/mkgray/web-growth.html.

[32] J. Cho and H. Garcia-Molina, “The evolution of the web and implications for an incremental crawler”, In Proc. of the 26th International Conference on Very Large Databases, Sep. 2000

[33] Junghoo Cho, Hector Garcia-Molina, “Parallel Crawlers”, WWW2002, May 7-11, 2002, Honolulu, Hawaii, USA.

[34] A. K. Sharma, J. P. Gupta, “Design of a Parallel Crawler based on Augmented Hypertext Documents (PARCAHYD)”, Ph.D. Thesis, HIT & M, Gwalior, Aug. 2003.

[35] S. Chakrabarti, M. van den Berg, and B. Dom, “Focused Crawling: A New Approach to Topic-Specific Web Resource Discovery” Computer Networks, 31(11-16): 1623-1640, 1999.

[36] M. Diligenti, F. Coetzee, S. Lawrence, C. L. Giles, and M. Gori, “Focused Crawling Using Context Graphs” In Proc. of VLDB, pages 527-534, 2000.

[37] Paolo Boldi, Bruno Codenotti, Massimo Santini, and Sebastiano Vigna, “Ubicrawler: A scalablefully distributed web crawler”, Proc. of Australian World Wide Web Conference (AusWeb), 2002

[38] J. Madhavan, D. Ko, L. Kot, V. Ganapathy, A. Rasmussen, A. Halevy , “Google’s Deep-Web Crawl”, In proceedings of Very large data bases VLDB endowment, pp. 1241-1252, Aug. 2008.

[39] UIUC, “http://metaquerier.cs.uiuc.edu/repository/datasets/bamm/index.htm , Web repository.

[40] Komal Kumar Bhatia, A.K. Sharma, “AKSHR: A Novel Framework of a Domain-specific Hidden Web Crawler”, IEEE International Conference of Advanced Computing 2009

[41] Rosy et al., “A Framework for Incremental Hidden Web Crawler”, (IJCSE) International Journal on Computer Science and Engineering, Vol. 02, No. 03, 2010, 753-758.

[42] Gustavo et .al.,” Automatic Filling of Hidden Web Forms: A Survey”, ACM SIGMOD Record, Volume 44 Issue 1, March 2015

[43] W. Wu, C. Yu, A. Doan, and W. Meng. An interactive clustering-based approach to integrating source query interfaces on the deep web. In *SIGMOD*, 2004

[44] Musen, M.A. (1992). “Dimensions of knowledge sharing and reuse”, Computers and Biomedical Research 25, pp. 435-467

[45] Dinesh Sharma, Komal Kumar Bhatia, A.K.Sharma,"Extraction of the Hidden Web Pages Through Single Attribute Forms with Automated Discovery Process", International Conference of Information Technology (ICIT-2007)

[46] Gruber, T.R. (1993). “A Translation Approach to Portable Ontology Specification Knowledge Acquisition 5”, pp. 199-220.

[47] Natalya F. Noy and Deborah L. McGuinness, “Ontology Development 101: A Guide to Creating Your First Ontology”, Stanford University, Stanford, CA.

[48] Cheng Sheng, Nan Zhang, Yufei Tao, Xin Jin : “Optimal Algorithms for Crawling a Hidden Database in the Web” in proceedings of the VLDB Endowment (PVLDB), 5(11): pno. 1112-1123, 2012.

[49] Protégé, http://protege.stanford.edu/.

[50] Resource Description Framework (RDF). http://www.w3.org/RDF/.

[51] OWL Web Ontology Language Reference. http://www.w3.org/TR/owl-ref/.

[52] SPARQL Query Language for RDF. http://www.w3.org/TR/rdf-sparql-query/.

[53] René Bernardo et. al. “Developing Ontology Systems as a Base of an Environmental Quality Management Model in México”, Journal of Environmental Protection, 2015, 6, 1084-1093 Published Online September 2015 in SciRe.

[54] Z. Zhang, B. He, and K. Chang. Understanding Web Query Interfaces: Best-Effort Parsing with Hidden Syntax. In SIGMOD Conference , 2004.

[55] Feng Zhao, Jingyu Zhou, Chang Nie, Heqing Huang, Hai Jin, “Smart Crawler: A Two-stage Crawler for Efficiently Harvesting DeepWeb Interfaces”, IEEE Transactions on Services Computing Volume: PP Year: 2015.

[56] Alexandros Ntoulas,” Focused crawling for the hidden web”, Springer Journal World Wide Web Volume 19, Issue 4, 2016.

[57] L.Reyes et al ,” Ontological models for information retrieval of product-service: Trends and open issues”, Proceedings of the 4th Spanish conferece on Information retrieval,June 14-16,2016

[58] A. Stolz, B. Rodriguez-Castro, A. Radinger, and M. Hepp. Pcs2owl: A generic approach for deriving web ontologies from product classification systems. In The Semantic Web: Trends and Challenges, pages 644--658. Springer, 2014.

[59] Deng, X. B., Ye, Y. M., Li, H. B., & Huang, J. Z. (2008). An Improved Random Forest Approach For Detection Of Hidden Web Search Interfaces. In Proceedings of the Seventh International Conference on Machine Learning and Cybernetics, Kunming, China.

[60] Rosy Madaan, Ashutosh Dixit, A.K. Sharma, Komal Kumar Bhatia, " Framework for Incremental Domain-specific Hidden Web Crawler", Proc. of third International Conference on Contemporary Computing (IC3), Jaypee Institute of Information Technology and University of Florida,May2010, published in Springer, LCNS.

[61] Ying wang," Automatic filling forms of deep web entries based on ontology" 2009 International conference on web information systems and mining.

[62] Uthayan, K. R., and G. S. Anandha Mala. "Hybrid Ontology for Semantic Information Retrieval Model Using Keyword Matching Indexing System." The Scientific World Journal 2015 (2015).

[63] R. Deepa, R Manicka Chezian An Involuntary Data Extraction And Information Summarization Expending Ontology International Journal of Applied Engineering Research, ISSN 0973-4562 Vol. 10 No.44 (2015)

[64] Davis Marques, "A Survey of Recent Research in Ontology Mapping", Simon Fraser University, School of Interactive Arts & Technology, 2005.

[65] Marc Ehrig and Steffen Staab, "QOM - Quick Ontology Mapping" ,The Semantic Web, ISWC, Volume 3298, 2004

[66] Wang, Y., Liu, W., Bell, D, "A concept hierarchy based ontology mapping approach", In: Bi, Y., Williams, M.-A. (eds.) KSEM 2010. LNCS, vol. 6291, pp. 101–113. Springer, Heidelberg (2010)

[67] Wordnet : https://wordnet.princeton.edu/

[68] Lin, D, "An information-theoretic definition of similarity", In: Proceedings of the 15th International Conference on Machine Learning (ICML'98), pp. 296–304 (1998)

[69] M. E. Okasha "Exploiting Ontology for Retrieving Data Behind Searchable Web Forms", ©2009 IEEE, Dept. of Computers and Systems, Mansoura University, Egypt

[70] Noy, N.F. and Musen, M.A. (2000). PROMPT: "Algorithm and Tool for Automated Ontology Merging and Alignment", In: Proceedings of the Seventeenth National Conference on Artificial Intelligence (AAAI-2000), Austin, TX

[71] An Hai Doan, Jayant Madhavan, Pedro Domingos, and Alon Halevy, "Learning to Map between Ontologies on the Semantic Web", In The Eleventh International WWW Conference, Hawaii, US, 2002.

[72] Alexander Maedche, Boris Motik, Nuno Silva, and Raphael Volz. Mafra - "a mapping framework for distributed ontologies", In Proceedings of the EKAW 2002, 2002

[73] Prasenjit Mitra, Gio Wiederhold, and Martin Kersten, "A graph-oriented model for articulation of ontology interdependencies", Lecture Notes in Computer Science, 1777:86+, 2000.

[74] Jian, N., Hu, W., Cheng, G., and Qu, Y. (2005). FalconAO: "aligning ontologies with Falcon. In Proceedings of K-CAP Workshop on Integrating Ontologies", pp. 85–91

[75] Shvaiko, P., & Euzenat, J. (2013), "Ontology Matching: State art and Future Challenges", IEEE , pp.1-15. Li W, Clifton C, Liu S (2000) Database integration using neural network: implementation and experiences. Inf Syst 2(1): 73-96.

[76] Wei Ding, Changshang Zhou and Na Yang, "Double Indexing Mechanism of Search Engine based on Campus Net", Proceedings of the 2006 IEEE Asia-Pacific Conference on Services Computing (APSCC'06), 2006.

[77] Ajit Kumar Mahapatra1, Sitanath Biswas2, "Inverted indexes: Types and techniques", IJCSI International Journal of Computer Science Issues, Vol. 8, Issue 4, No 1, July 2011 ISSN (Online)

[78] Fabrizio Silvestri, RaffaelePerego and Salvatore Orlando, "Assigning Document Identifiers to Enhance Compressibility of Web Search Engines Indexes", In the proceedings of SAC,2004

[79] O. Zamir, O. Etzioni, O. Madanim, and R.M. Karp, "Fast and Intuitive Clustering of Web Documents", Proc. Third Int'l Conf. Knowledge Discovery and Data Mining, pp. 287-290, Aug. 1997

[80] Parul Gupta, Dr. A.K.Sharma, "Context based Indexing in Search Engines using Ontology", International Journal of Computer Applications (0975 – 8887) Volume 1 – No. 14,2010.

[81] Pooja et.al. , "A New Approach for Context Based Indexing in IR System Using BST ", International Journal of Engineering Trends and Technology (IJETT) – Volume 11 Number 8 - May 2014

[82] Nidhi Tyagi & Jain Anchal, "Context Based Web Indexing For Semantic Web" IOSR Journal of Computer Engineering (IOSR-JCE) e-ISSN: 2278-0661, p- ISSN: 2278-8727Volume 12, Issue 4 (Jul. - Aug. 2013), PP 89-93 www.iosrjournals.org

[83] Larry Page, and Sergey Brin, Rajeev Motwani, Terry Winograd, 'The PageRank Citation Ranking: Bring Order to the Technical report in Stanford University, 1998

[84] Wenpu Xing and Ghorbani Ali, ‘Weighted PageRankAlgorithm’, Proceedings of the Second Annual Conference on Communication Networks and Services Research (CNSR ’04), IEEE, 2004

[85] N. Batra, A. Kumar, Dr. D. Singh and Dr. R.N. Rajotia “Content Based Hidden Web Ranking Algorithm (CHWRA)”, Advance Computing Conference (IACC), 2014 IEEE International, 2014 IEEE, DOI 10.1109/IAdCC.2014.6779390 Page(s): 586 – 589.

[86] G.Kumar; N. Duhan; A.K. Sharma, ‘Page Ranking Based on Number of Visits of Links of Web Page ‘, International Conference on Computer & Communication Technology, (ICCCT), 2011

[87] Babita Ahuja , Dr. Anuradha, “SCUM: A Hidden Web Page Ranking Technique”, International Journal of Innovative Research in Advanced Engineering (IJIRAE) ISSN: 2349-2163 Volume 1 Issue 10 (November 2014).

[88] Brian Wai Fung Wong’s, “Deep-Web Search Engine Ranking Algorithm”, MIT

[89] Saravanan Thirumuruganathan, Nan Zhang, Gautam Das, “Rank Discovery From Web Databases” by University of Texas at Arlington; George Washington University published in Proceedings of the VLDB Endowment, Vol. 6, No. 13Copyright 2013 VLDB Endowment 21508097/13/13.

[90] Raju Balakrishnan’s, “Trust and Profit Sensitive Ranking for the Deep Web and On-line Advertisements ”, Arizona State University August 2012.

[91] thesaurus : http://www.thesaurus.com/

[92] Jena API Tutorial http://jena.sourceforge.net/tutorial/RDF_API/.

[93] Manvi, Ashutosh Dixit, Komal Kumar Bhatia, Bhumika Wadhwa, “Generating domain specific ontology for retrieving hidden web data”, IEEE International Conference ISCON,2014, GLA University, Mathura,1-2 March 2014.

[94] Manvi, Komal Kumar Bhatia and Ashutosh Dixit, “Automatic Generation of Ontology from Hidden Web Pages”, 50th Golden jubilee international annual convention of Computer Society of India (CSI-2015) theme Digital Life, organised by BVICAM New Delhi, 2nd to 5th December 2015.

[95] Manvi, Komal Kumar Bhatia and Ashutosh Dixit, “A novel design of hidden web crawler using ontology”, International Journal of Engineering Trends & Technology (IJETT), August 2015, ISSN: 2231-5381 DOI: 10.14445/22315381/IJETT-V26P204. DBLP indexed.

[96] Manvi, Ashutosh Dixit, Komal Kumar Bhatia, Rajiv Mishra, “A Novel Architecture of Ontology Mapping System for Hidden Web Retrieval", IEEE International Conference (ICROIT-2014), on Feb 6-8, 2014.at Manav Rachna International University Faridabad.

[97] Manvi, Komal Kumar Bhatia and Ashutosh Dixit, “Ontology based Indexing Technique for Hidden Web”, 2nd International conference on Recent Development in Sciences Engineering and Technology organised by GD Goenka University, Guragaon, 30-31, 2015.

[98] Manvi, Jyoti Yadav, “Design of a novel Ranking Technique for Hidden Web pages”, in Advances in Computer Science and Information Technology (ACSIT), Volume 2, Number 9; April-June, 2015 pp. 48-51.

[99] Jianguo Lu, “Ranking Bias in Deep Web Size Estimation Using Capture Recapture Method”, School of Computer Science, University of Windsor, Canada.

[100] DatabasLayout,http://jena.sourceforge.net/DB/index.html, (as of02/24/2005).Jena API.

[101] Manvi, Jyoti Yadav, “Design of a novel Ranking Technique for Hidden Web pages” in Advances in Computer Science and Information Technology (ACSIT), Volume 2, Number 9; April-June, 2015 pp. 48-51.

[102] Manvi, Komal Kumar Bhatia and Ashutosh Dixit “Annotating Ranking Techniques for Hidden Web: A Review” in Journal of Network Communications & Emerging Technologies (JNCET), Volume 5, Special Issue 2, December (2015).

[103] Swoogle : http://swoogle.umbc.edu/

[104] A. Gangemi, C. Catenaccia, M. Ciaramita, and J. Lehmann. “Modelling ontology evaluation and validation”, In Y. Sure and J. Domingue, editors, Proceedings of the 3rd European Semantic Web Conference (ESWC2006), number 4011 in LNCS, Budva, Montenegro, June 2006. Springer-Verlag.

[105] A. Gangemi, C. Catenacci, M. Ciaramita, and J. Lehmann. “Ontology evaluation and validation: an integrated formal model for the quality diagnostic task”, Technical report, Laboratory of Applied Ontologies – CNR, Rome, Italy, 2005. available at http://www.loacnr.it/Publications.html.

[106] PORTER’S ALGORITHM : https://www.eecis.udel.edu/~trnka/CISC889-11S/lectures/dan-porters.pdf

[107] Ping Wu, Ji-Rong Wen, Huan Liu, Wei-Ying Ma :“Query Selection Techniques for Efficient Crawling of Structured Web Sources” in proceedings of the 22nd

International Conference on Data Engineering (2006), Pages 47, IEEE Computer Society, Washington

[108] Sergey Melink, Hector Garcia-Molina, Erhard Rahm. Similarity Flooding: A Versatile Graph Matching\ Algorithm and its Application to Schema Matching. In Proc. 18th International.Conf. On Data Engineering, San Jose CA, 2002

BRIEF PROFILE OF RESEARCH SCHOLAR

Manvi is pursuing her Ph.D in Computer Engineering from YMCA University of Science and Technology, Faridabad .She did her M.Tech.(Computer Engineering)from YMCA University of Science and Technology in year 2008,and B.Tech.(Computer Science) from Kurukshetra University, Kurukshetra in 2005. Ms.Manvi has over 10 years of experience in teaching B.Tech ,MCA and M.Tech courses. Her areas of interests includes Information Retrieval, Programming Languages(C,C++ etc.),Data structures and Web Technologies. She has published 14 research papers in various journals and conferences of international fame. She is a recipient of Best paper Award in 2nd International conference on Recent Development in Sciences Engineering and Technology organised by GD Goenka University, Gurgaon. Currently, she is working as Assistant Professor in the department of Computer Engineering at YMCAUST, Faridabad.

LIST OF PUBLICATIONS OUT OF THESIS

(i) List of Published Papers

S.No	**Title of the paper**	**Name of Journal where Published**	**No. Volume &**	**Issue**	**Year**	**Pages**
1	A novel Technique for creating Semantic database for hidden web	International Journal of Innovative research in Science & Technology(IJIRST	Volume 1	Issue 11	2015	299-305
2	Design and Implementation of Domain based Semantic Hidden Web Crawler	International Journal of Innovations and Advancement in Computer Science	Volume 1	Special Issue	2015	2347-8616
3	A novel design of hidden web crawler using ontology	International Journal of Engineering Trends & Technology (IJETT),	Volume 26	Issue 2	2015	204
4	Design of a novel Ranking Technique for Hidden Web page	Advances in Computer Science and Information Technology (ACSIT),	Volume 2	Issue 1	2015	48-51.
5	Annotating Ranking Techniques for Hidden Web: A Review	in Journal of Network Communications & Emerging Technologies	Volume 5	Special Issue 2	2015	

LIST OF PUBLICATIONS

INTERNATIONAL JOURNALS

1. Manvi, Ashutosh Dixit, Komal Bhatia and Rinki Kardam, “A novel Technique for creating Semantic database for hidden web”, in International Journal of Innovative research in Science & Technology(IJIRST), Volume 1,Issue 11,ISSN(online):2349-6010,April 2015, pp 299-305.
2. Manvi, Ashutosh Dixit, Komal Bhatia and Jyoti Yadav, “Design and Implementation of Domain based Semantic Hidden Web Crawler” in volume 4, Special Issue, May 2015 of International Journal of Innovations and Advancement in Computer Science ISSN 2347-8616. **DBLP indexed.**
3. Manvi, Komal Kumar Bhatia and Ashutosh Dixit, “A novel design of hidden web crawler using ontology”, in International Journal of Engineering Trends & Technology (IJETT), August 2015, ISSN: 2231-5381 DOI: 10.14445/22315381/IJETT-V26P204. **DBLP indexed.**
4. Manvi, Jyoti Yadav, “Design of a novel Ranking Technique for Hidden Web pages” in Advances in Computer Science and Information Technology (ACSIT), Volume 2, Number 9; April-June, 2015 pp. 48-51.
5. Manvi, Komal Kumar Bhatia and Ashutosh Dixit “Annotating Ranking Techniques for Hidden Web: A Review” in Journal of Network Communications & Emerging Technologies (JNCET), Volume 5, Special Issue 2, December (2015).

INTERNATIONAL CONFERENCES

1. Manvi, Ashutosh Dixit, Komal Kumar Bhatia, “Design of an Ontology based Adaptive Crawler for Hidden Web” In the proceedings of **IEEE** International Conference on Communication Systems and Network Technologies (CSNT-2013) Apr 6-8, 2013 at MIR Labs Gwalior India pp 659-663. **Print ISBN:** 978-1-4673-5603-9. **(Scopus Indexed)**
2. Manvi, Ashutosh Dixit, Komal Kumar Bhatia, Rajiv Mishra, “A Novel Architecture of Ontology Mapping System for Hidden Web Retrieval", **IEEE** International Conference (ICROIT-2014), on Feb 6-8, 2014.at Manav Rachna International University Faridabad. **(Scopus Indexed)**

3. Manvi, Ashutosh Dixit, Komal Kumar Bhatia, Bhumika Wadhwa, “Generating domain specific ontology for retrieving hidden web data”, **IEEE** International Conference ISCON,2014, GLA University, Mathura,1-2 March 2014. **(Scopus Indexed).**
4. Manvi, Komal Kumar Bhatia and Ashutosh Dixit, “Automatic Generation of Ontology from Hidden Web Pages”, 50th Golden jubilee international annual convention of Computer Society of India (CSI-2015) theme Digital Life, organised by BVICAM New Delhi, 2nd to 5th December 2015. **Conference proceedings published by Springer**.
5. Manvi, Komal Kumar Bhatia and Ashutosh Dixit, “Ontology based Indexing Technique for Hidden Web”, 2nd International conference on Recent Development in Sciences Engineering and Technology organised by GD Goenka University, Guragaon, 30-31, 2015. Proc published with ISBN no. 978-93-84869-85-4. (**This paper received best paper award)**

NATIONAL CONFERENCES

1. Manvi, Ashutosh Dixit, Komal Kumar Bhatia, “Deep web annotation through hidden web crawlers: a review”, SIM 2012 at YMCAUST, Faridabad 3- 4th Dec, 2012. Indexed on International Journal of Multidisciplinary Research Studies, Dec 2012.
2. Manvi, Pramesh Dahiya, "Issues and Challenges in Design of an effective Search Engine", Contemporary Issues in Commerce, Economics, Management and IT held on 15-16 November, 2013 at Govt. College, Jind.

ACHIVEMENTS

1. **Won Best Paper Award for the paper** “Ontology based Indexing Technique for Hidden Web”, at 2nd International conference on Recent Development in Sciences Engineering and Technology organised by GD Goenka University, Gurgaon.