DYANAMIC QUERY PROCESSING FOR HIDDEN WEB DATA EXTRACTION

THESIS

submitted in fulfillment of the requirement of the degree of

DOCTOR OF PHILOSOPHY

to

J.C. BOSE UNIVERSITY OF SCIENCE & TECHNOLOGY, YMCA

by

BABITA AHUJA

Registration No: Ph.D.-02-2K12

Under Supervision of

Dr. ANURADHA
ASSISTANT PROFESSOR
J.C. Bose University of Science & Technology, YMCA, Faridabad

Dr. DIMPLE JUNEJA
DEAN (Research & Development)
Poornima University, Jaipur

Department of Computer Engineering

Faculty of Engineering and Technology

J.C. Bose University of Science & Technology, YMCA, Faridabad

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

February 2019

DEDICATION

Dedicated to my husband….

DECLARATION

I hereby declare that this thesis entitled **"DYNAMIC QUERY PROCESING FOR HIDDEN WEB DATA EXTRACTION"** by **BABITA AHUJA**, being submitted in fulfilment of requirement for the award of Degree of Doctor of Philosophy in the Department of Computer Engineering under Faculty of Engineering and Technology of J.C. Bose University of Science and Technology, YMCA, Faridabad, during the academic year March 2013 to February 2019, is a bonafide record of my original work carried out under the guidance and supervision of **Dr. Anuradha, Assistant Professor, Department of Computer Engineering, J.C. Bose University of Science and Technology,YMCA, Faridabad and Dr. Dimple Juneja, Dean, Research and Development, Poornima University, Jaipur** 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.

BABITA AHUJA

Ph.D-02-2K12

CERTIFICATE

This is to certify that this thesis entitled **"DYNAMIC QUERY PROCESSING FOR HIDDEN WEB DATA EXTRACTION"** by **BABITA AHUJA,** 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 J.C. Bose, University of Science and Technology, YMCA, Faridabad, during the academic year March 2013 to February 2019, is a bonafide 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.

More than two research papers in prestigious refereed journals have been published and all other academic requirements are fulfilled.

The thesis is based on the individual, original work of the candidate, which is previously unpublished research work and the thesis does not contain any material that infringes the copyright of any other individual or organization and does not hurt the sentiments of any individual(s) or religion(s). The information such as text, tables, equations, diagrams, figures, charts, and photographs taken from sources such as published work, like research papers, books, periodicals, web sites and other resources has been cited appropriately. Further, the opinions expressed or implied in the thesis are entirely of the candidate.

Dr. ANURADHA
ASSISTANT PROFESSOR
J.C. Bose University of Science & Technology, YMCA, Faridabad

Dr. DIMPLE JUNEJA
DEAN (Research & Development)
Poornima University, Jaipur

ACKNOWLEDGEMENT

I would first like to thanks my supervisors Dr. Anuradha, Assistant Professor, Department of Computer Engineering, J.C. Bose University of Science and Technology, YMCA, Faridabad and Dr. Dimple Juneja, Dean, Research and Development, Poornima University, Jaipur for their mentoring and guidance during this research work. They have not only inspired, guided and supported but also made my dream of pursuing and completing the research work true. I am grateful to both of them for providing a cultivating environment that ropes liberty, open consultations, and constructive criticism, without which this work would not have been possible. They have always encouraged me to explore innovative things and were always there to guide me for improvisation by giving their invaluable suggestions. They have taught me to be strong and push forward when it is hard. I thank both of them from all my heart for the help in all aspects of my Ph.D. I am indebted to the long 'wisdom talks', which will benefit me throughout my future. I would like to express my sincere gratitude to Prof. Komal Bhatia and Prof. Atul Mishra, Professors, Department of Computer Engineering, J.C. Bose University of Science and Technology, YMCA, Faridabad, for trusting and believing in me. Both of them inspired me to set the research standards high and not to compromise with them. They have cheered for me during my ups and cheered me up during my downs. I am extremely honoured to have worked and interacted with both of them. I am also thankful to all my students, faculty and staff members of the department who helped me directly or indirectly in completing my work. I would like to thank my husband, Ashish Ahuja for inspiring me and showing me that there are no limits and I can pursue whatever I am passionate about. He is the most amazing person I have met and I am very blessed to have him near me. He constantly cheered me when I was down and helped me when I needed advice and aid on my research.

Finally, I would like to thank my supportive and affectionate family for their constant support. I am grateful to my father and mother for a wonderful education, my sister, and my brother-in-law for supporting my education.

(BABITA AHUJA)

ABSTRACT

WWW has become an integral part of everyone's life. Gone are the days when people referred books, newspapers or any non-digital medium to find information about any topic. Now a days the non-digital medium of information has been replaced by the digital medium such as e-books, e-banking, websites etc. The users are now able to access the digital information by the help of internet and WWW using mobile phones, tablets, laptops, desktops etc. WWW has a collection of web documents which are connected to each other by the help of hyperlinks. WWW runs on internet and allows users to access web documents stored on different computers that are interconnected to each other. The search engines have been developed for the extraction of web documents from WWW.

The web documents lying on WWW can be classified as hidden web and surface web. The web documents from surface web are indexable as well as crawlable by the search engines easily and hence they can be displayed to users as per their input query. In contrast to this, hidden web documents are neither indexable nor crawable by the traditional search engines. So web documents from hidden web are not shown when users issue their query. The few reasons behind the invisibility of hidden web are disconnected URL's, no-index tag, user authentication, web form processing etc. According to the study conducted by various researcher the data from hidden web is of very high quality and quantity therefore extraction of data from hidden web is very important to improve the performance and the result quality of search engines.

As the data of the hidden web cannot be crawled and indexed by the conventional search engines, the enhancement of these search engines has become very crucial and challenging. A major part of hidden web data is accessible through query interfaces. Users fill the query interfaces with their desired input fields and submit them. The data is then fetched dynamically from the hidden web databases and result pages are returned. The crawlers are not able to process the web forms and access data behind them. The hidden web crawlers should be developed which extracts the data from these hidden web databases and uncover this big part of WWW. The data of a particular domain is present on both the surface web as well as the hidden web and users have to fetch data from these sections separately, hence there is a need for a system which integrates and processes the data form both surface and hidden web.

In this work, a novel framework has been proposed which aims to solve the challenges highlighted above. The proposed solution, Dynamic Query Processing for Hidden Web Data Extraction (DQPHDE) extracts data from hidden web and integrates data from surface and hidden web to meet user's requirements. Development of the proposed work follows the parallel working of different modules namely Clustering using Graph and Relationship Extraction, Semantic Based Text Mining and Summarization (STMS), Web Form Extraction and Ranking (WFER) and Expert System for Data Region Extraction and Integration (ESDREI). A novel graph based clustering has been proposed in CGRE for clustering the web documents. For the purpose of integration of unstructured text from web documents and creating summary of the text, new text mining approach has been developed in STMS. In WFER, a new fuzzy rule based expert system has been proposed for the identification and ranking of query interfaces. ESDREI has used fuzzy expert system to extract the records from the result pages. Using the proposed technique the data from academic domain has been extracted and an academic search engine prototype has been developed. In the developed academic search engine, all kinds of data such as call for papers, abstracts, research articles, books etc. needed by the student or researcher are extracted from both surface web as well as hidden web and is returned to the user at a single place. The results of the proposed work are compared with the already existing techniques and the results obtained are promising.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1. INTRODUCTION

WWW is a largest source of digital information. It has a diverse range of information about studies, fashion, politics, tourism, social networking, mail systems, vehicles, business, sports, cooking, countries, history, illegal activities, drugs etc. Thus WWW has become a very essential part of every one's life. The internet has been used by more than 50% of the population and it is believed to be having over 4157 billion users in 2017 [1]. According to Internet Live Stats [2], a website of the international Real Time Statistics Project, every second approximately 7986 tweets are tweeted, more than 66418 Google queries are searched and more than 2 million emails are sent. This just gives a hint about the pace of growth of WWW. As the WWW is growing, so is the problem of managing and searching the information in it. In order to help the end users find the desired information from this bulk of information various search engines have been developed like Google, Bing, Yahoo, Ask.com, AOL.com etc. Some search engines are designed to extract the unstructured data, while others extract the structured data from WWW [3]. The information about a topic on WWW is scattered across different types of web pages such as surface web pages which mostly contain the unstructured web pages, hidden web pages which mostly contain the structured web pages and file formats etc. It has become very difficult for the users to filter the desired information from different sections of WWW because of factors such as size of results fetched, redundancy of data and the data is hidden behind query interfaces. The users have to go through every page, process it and then only they are able to extract the desired information. The most promising solution is the integrated system, where users can enter query and get all the processed data from different parts of WWW hence saving a lot of their time.

Conventionally, in most of the search engines when the user enters the query, the result pages returned may contain the surface web information, some files and some query interfaces. The user has to extract the desired information from the result web pages. In order to extract data behind the query interfaces user has to fill and submit the query form of all the result pages. Thus there is a need for a system which processes the

surface web as well as hidden web data automatically and will returns integrated results to the user. Although there are many hidden web data extractors, however to the best of our knowledge and understanding, none of them have tried to use the artificial intelligence approach. Hence "Dynamic Query Processing for Hidden Web Data Extraction" (DQPHDE) is being proposed in this research that exploits the artificial intelligence technique, extracts the data behind the query interfaces, processes and then integrates the information from different sections of WWW. Thus the motivation of DQPHDE is to provide processed and integrated results from both the surface web as well as hidden web.

1.2. TRADITIONAL SEARCH ENGINES

Search engine is a software program that is designed to search the Internet which is one of the widely used and largest information retrieval system [4]. It consists of tremendous amount of web sites. A lot of tools, directories and search engines have been created to extract the data from the internet. These tools are based on FPT protocol, Gopher protocol etc.

1.2.1. Evolution of Search Engines

The information systems were developed in 1960's and later search engine were developed. Search Engine is known as the largest information retrieval system. The evolution of information systems and search engines has been discussed below.

- Magic Automatic Retriever of Text: Magic Automatic Retriever of Text was developed at Cornell University in 1960's by Gerard Salton [10] and his team. Gerard Salton has been known as the founder of Information Systems. The techniques such as vector space model, inverse document frequency, term frequency used in information retrieval has been developed by Gerard [11] and these features were included by Gerard and his team in Magic Automatic Retriever of Text.
- Archie: Archie was a tool developed in 1990 by Alan Emtage at McGill University in Montreal. Archie was developed for indexing the FTP documents and helped people to find specific files [12]. Archie was developed when internet was developed whereas WWW was not. Archie was only developed with the aim of exchanging files between computers using internet.

- Veronica: Veronica was developed in 1992 by Steven Foster and Fred Barrie at the University of Nevada, Reno [13]. Veronica was a search engine for indexing the documents stored on thousands of gopher servers in the world that is also known as gopherspace.
- Aliweb: Aliweb was developed in 1993 by Martijn Koster while working at Nexor [14]. Aliweb stands for “Archie Like Indexing for the Web”. Aliweb was considered as the first web search engine as Archie and Veronica were just the indexers. Aliweb created directories for web only.
- Search Directories: In 1994 the first browsable web directory “Tradewave Galaxy” was created. Tradewave galaxy had the categories and the subcategories of files. Later Yahoo directory was built in 1994 by David Filo’s and Jerry Yang’s at Stanford University. It was the most common way to find website by browsing the list and the websites were organised as topics.
- AltaVista: AltaVista was the search engine which was developed in 1995 [15]. It was one of the most widely used search engine. AltaVista was developed by Paul Flaherty and his team at Digital Equipment Corporation's Network Systems Laboratory and Western Research Laboratory. It was the first to adopt natural language queries as well as Boolean search techniques. And to aid in this, it was the first to offer "Tips" for good searching prominently on the site.
- Google: Google was developed in March 1998 by Larry Page and Sergey Brin, students of Stanford University [16]. Today google is the most popular and widely used search engine.

A lot of search engines have been developed after that such as Ask, Msn, and Bing etc. as shown in Fig 1.1.

Figure 1. 1. Popular Search Engines.

1.2.2. Search Engines Features

The main objective of a search engine is to search and structure the distributed and unstructured data found on the Internet [6]. Its main features are presented below:

1. **Competency in searching**: The efficiency of the search engine depends upon the results returned to the end user's query. There are different search facilities for improving the efficiency of search engine for e.g. phrase searching, truncation and limiting facilities.
2. **Evaluation measures**: The technique of extracting information is measured by considering the three basic information retrieval measures i.e., precision, recall and response time.
3. **Content Quality**: The quality of output of search engines depends upon two factors. The first one is how many output options are provided by the search engines. The second factor determining the quality of the content are the relevant result pages. The architectural composition is the major factor impacting the output content provided by a search engines.
4. **User friendly Interface**: End users are able to interact with the search engines by the help of user interface only. How easily a user is able to understand the working and features of search engine increases the quality of the search engine from the user's perspective. So the user interface should be friendly and comfortable.

Table 1.1 shows the tabular comparison of the popular five search engines [8] based on various features.

Table 1. 1. Comparison of Different Search Engines based on Features.

Features	Google	Yahoo	AltaVista	Ask	Bing
Search Web	Yes	Yes	Yes	Yes	Yes
search images	Yes	Yes	Yes	Yes	Yes
Search Videos	Yes	Yes	Yes	Yes	Yes
Search News	Yes	Yes	Yes	Yes	Yes
Search Maps	Yes	No	No	No	Yes
Search Books	Yes	No	No	No	No
Advance Search	Yes	Yes	Yes	Yes	Yes
Change Background	Yes	No	No	Yes	Yes
Change search Settings	Yes	Yes	Yes	Yes	Yes
Display no. of Results	Yes	Yes	Yes	No	Yes
Shopping	No	Yes	Yes	Yes	Not Given
Translation Services	Yes	No	Yes	No	Yes
Multi Language Support	Yes	No	No	No	No
Questions/answers	No	Yes	Yes	Yes	No
Directory	Yes	Yes	Yes	No	No
Advertising Programs	Yes	Yes	Not Given	No	No
Business Solutions/Services	Yes	No	Yes	No	No
Themes	No	No	Yes	No	No
Case sensitive	No	No	No	No	No
Finance	Yes	Yes	No	No	No
Safe Search	Yes	Yes	Yes	Yes	Yes
Search pad	No	Yes	Yes	No	No
Careers	No	Yes	No	Yes	No
Preferences	Yes	Yes	Yes	Yes	Yes

Studies have revealed that the three most widely used web search engines and their approximate share of usage as of late 2010, is Google 91%, Yahoo 4%, Bing 3% and others 2% [9].

1.3. DIGITAL ACADEMIC SERACH ENGINES

A lot of hidden web data extraction tools have been developed for different domains such as people information extraction, government federal information, geological surveys, medications, government documents, academic research documents. In this research the academic research documents have been taken. A lot of tools have been developed to extract hidden academic research data such as DOAJ, BASE, CORE, research gate etc. The idea is to organize and ease the information retrieval process for the researcher. The details of these tools are given below.

1. DOAJ (Directory of Open Access Journals)

The Directory of Open Access Journals (DOAJ) [17] is a website that lists open access journals and is maintained by Infrastructure Services for Open Access (IS4OA) [18]. The open access journals are the scientific and scholarly journals that are of high quality standards. Quality is find out by exercising peer review or editorial quality control. These open access journals use a funding model which does not charge for access of articles. The aim of DOAJ is to "increase the visibility and ease of use of open access scientific and scholarly journals thereby promoting their increased usage and impact". These are academic papers that are available to anyone "without financial, legal, or technical barriers other than those inseparable from gaining access to the Internet itself." In short, knowledge is free here.

2. CORE (Computing Research and Education)

CORE was developed by the combined efforts of computer science departments in Australia and New Zealand. CORE was formerly known by the name Computer Science Association, CSA [19]. CORE assists and advances research in computer science and information technology in higher education and research institutes. It provides a forum for those interested in computer science and information technology so as to stimulate discussion of relevant issues. It is an aggregation of the world's open access research papers. When the user issues the query, the list of research articles is shown to the user. CORE also analyses the text of the research articles and gives the option to the user to view similar articles.

3. BASE (Bielefeld Academic Search Engine)

The BASE is a multi-disciplinary search engine for scholarly internet resources, created by Bielefeld University Library in Bielefeld, Germany [124]. It is based on free and open-source software such as Apache Solr and VuFind [124]. BASE collects Open

Archive Initiative metadata from institutional databases and other academic digital search engines which uses the Open Archives Initiative Protocol for Metadata Harvesting (OAI-PMH). BASE then normalizes and indexes the data gathered for which later helps in searching. BASE also crawls and index the web pages present on the WWW. Using BASE the users can get bibliographic metadata information including abstracts [20] . BASE does not currently offer full text search.

4. Google Scholar

Unlike Google search engine, Google Scholar is an academic search engine. It crawls and indexes the academic documents mainly the research articles. The crawler of Google Scholar crawls the online journals available, the online university databases and other online sources for serving the user's needs [21]. The various filters are provided by Google scholar by the help of which the researchers can modify the output. This will give access to relevant documents.

5. Microsoft Academic

Microsoft Academic help the researchers by providing publications and literature free of cost. It was launched by Microsoft Research in 2016. The new data structures have been used for the storage and indexing. The searching here is done using the semantic search techniques. It currently indexes over 375 million entities, 170 million of which are academic papers [22].

6. Research Gate

Research gate was launched in 2008. Research gate includes many popular features and functionalities that are now common among social networking sites such as creating user profiles, posting public and private messages, sharing information and finding other users with similar interest [23].

7. Academia.edu

Academia.edu is one of the academic search engines launched in 2008 [5]. It stores the details of publications, helps the researchers to follow topics or other researchers of their interest. It also helps the researchers to publish the blogs [24]. The researchers can also find how many other researchers have visited their profile. From which countries the researchers followed and what they found interesting in one's profile can also be found here.

The comparison between the different digital academic libraries is shown in Table 1.2.

Table 1. 2. Comparison of Different Digital Academic Libraries.

	Integration of Hidden Web and Surface Web	**Social Networking**	**Clustering**	**Summarization**
DOAJ	No	No	Yes	No
CORE	No	No	Yes	No
BASE	No	No	Yes	No
Google Scholar	No	No	Yes	No
Microsoft Academic Research	No	No	Yes	Yes
Research Gate	No	Allow Messaging	Yes	No
Academia.edu	No	No	Yes	No

1.4. MOTIVATION

The accumulation of the ubiquitous amount of data on the web has inclined the researchers towards extracting the relevant content through search interfaces. The static and dynamic collection of documents encourages users to explore the hidden web in order to access the pages from different websites. According to recent studies, the content provided by many Hidden Websites is often of very high quality and can be extremely valuable to many users. The main reasons which catch the eye of the researchers for its extraction this type of data are given below.

1. **Size of hidden web and the rate of growth of hidden web**

The hidden web contains a vast amount of information. The size of the hidden web has increased at a very fast speed and it is still expanding. In 2000 the surface web contained nearly 2.5 billion documents and had a size of 19TB. According to the study of Bergman [25], hidden web has 550 billion web pages which is 220 times the number of web documents in the surface web. Bergman in his studies calculated the quality of hidden web and found out that it is 1000-2000 times better than surface web [25]. So in early 2000, hidden web was in its growing phase and very less percentage of it was

indexable by search engines of that time. The statistics of hidden web size done by Bergman are shown in Table 1.3.

Table 1. 3. Size of Hidden Web and Surface Web in 2000.

	Surface Web	Hidden Web
Documents	2.5 billion	550 billion
Size	19TB	7500TB

In 2005, the size and the quality of hidden web was analysed again by B. He [26]. The study showed that the size of hidden web has increased in five years. The websites were crawled by the crawler and the websites having hidden web contents were extracted. It was found that 307,000 hidden websites existed. These websites contained nearly 1,258,000 web forms. These web forms were linked to 450,000 web database servers. So the hidden web had grown 3 – 7 times larger during 2000-2005 [26], as shown in Table 1.4.

Table 1. 4. Size of Hidden Web and Surface Web in 2005

	Surface Web	Hidden Web
Documents	11.5 billion	1650 -3850 billion
Size	60TB – 100TB	22500 -52500TB

Then again after 5 years in 2011, the size and the quality of hidden web estimated by Kunder [27]. The surface web was found to have at least 14 billion documents [27]. The surface web was found to have 456TB of data. The size of the hidden web was increased so much that was not feasible to calculate the size of whole hidden web. But according to previous studies if the hidden web is still 400 – 500 times larger than the surface web, that means there are 8000 – 10 000 billion web pages and the size of hidden web will be nearly around 186 000 TB – 232 500 TB, as shown in Table 1.5. The growth of the hidden web and surface web from 2000 to 2011 is shown in Fig 1.2.

.Table 1. 5. Size of Hidden Web and Surface Web in 2011.

	Surface Web	Hidden Web
Documents	14 billion	8000 -10000 billion
Size	465 TB	186000- 232500TB

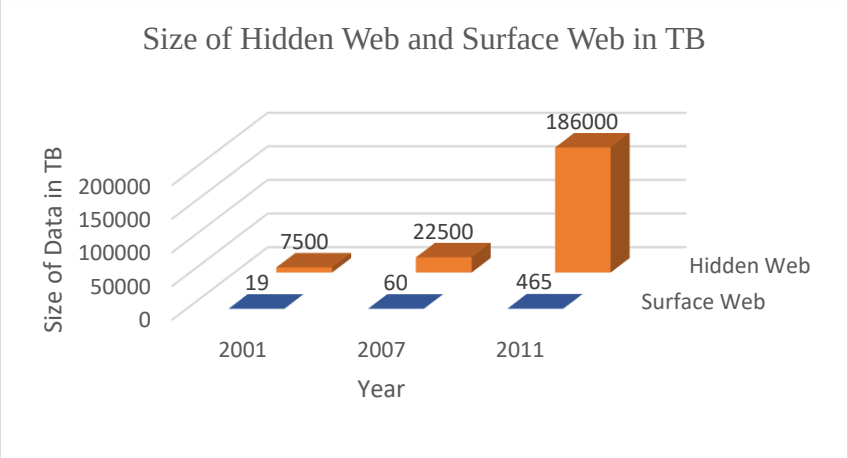

Figure 1. 2. The Growth of Hidden Web and Surface Web.

2. Most of the hidden web is inaccessible to the traditional search engines

The data behind query interfaces forms a major part of the hidden web. Most of the content of the hidden web is not indexed by the traditional search engines. According to a study published in Nature, Google indexes not more than 16 percent of the surface web and misses all of the hidden web [25]. So when query is issued on search engines then only .03% of information from WWW is returned [120]. This amount of information fetched is like if WWW is an ocean then we are getting only top two feet of ocean's content. So a large part of hidden web is non-indexable and is not accessable to the users. The hidden web has a different type of content such as online shopping data, chats, publications, databases etc. Bergman [25] calculated the percentage of the different type of content found in the hidden web as shown in Fig 1.3.

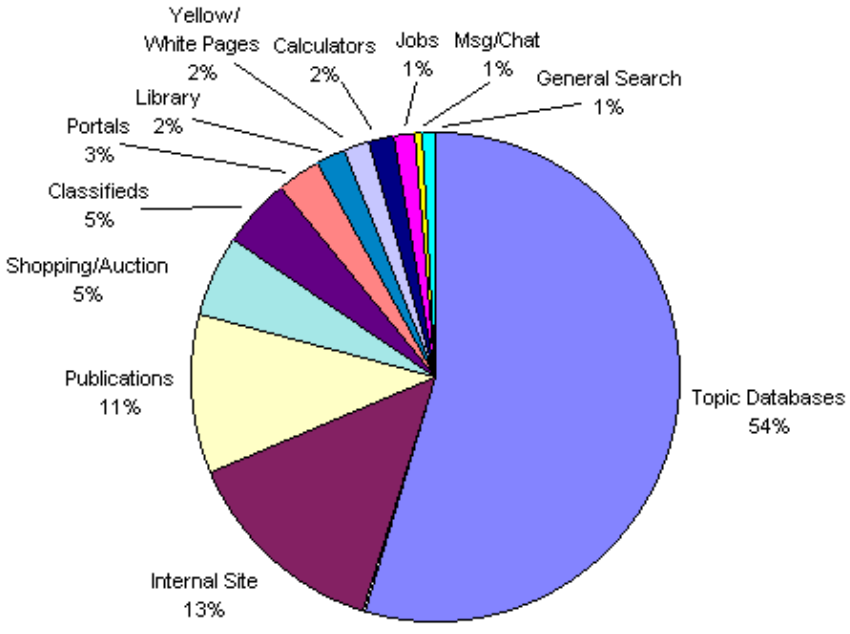

Figure 1. 3. Distribution of Hidden Web Site by Content.

3. **Lack of integrated data from both the surface web and hidden web**

The details of a particular web entity can be found on both the surface web and the hidden web. To extract the full details users have to browse both the surface web and hidden web. There is no single source of such integrated information.

4. **Lack of integrated and linked data from hidden web**

The details of an entity can be scattered on more than one hidden web source. Most of the hidden data extractors extract the information and place them in a single database. The information extracted is not processed, it is just aggregated. It may result in the duplication of information. There is no integrator which links the information from multiple sources and removes the redundant information.

1.5.OBJECTIVES OF RESEARCH WORK

On the basis of literature grilled during the initial phase of the research work, following objectives have been identified in the light of challenges stated in the above section.

- Develop an algorithm to cluster the web documents based on the type of content present in them.
- Design an algorithm to integrate the unstructured data from multiple web sources and summarize it.
- Develop an intelligent approach to automatically extract the web forms from the web pages and rank the query interfaces.
- Design an approach to intelligently extract the result sections from the web pages and create an integrated and linked database.
- Evaluation and comparison of proposed work with the existing techniques used for extracting hidden web.

1.6. ENABLERS AND BARRIERS TO ACHIEVE THE OBJECTIVES

Web crawlers are the software that downloads the publically indexable web pages and help the search engines. The major barriers and challenges in designing an efficient web crawler to crawl both the surface web and the hidden web contents are given below.

- **Heterogeneous types of web pages**

  Information about the entities is present on different types of web pages such as unstructured data, file formats, structured data, query interfaces etc. So in

order to integrate the information, all types of web pages should be processed.

- **Identification of the domains**

  There are large numbers of domains lying on WWW. In order to access the relevant information from the web pages the identification of domain is a big challenge.

- **Redundancy of Data**

  There are a lot of websites for a particular domain, so most of the information is redundant on the web. In order to extract the unique information user has to go through all the web pages.

- **Semantic Similarity of Sentences**

  As most of the information in WWW is redundant, it becomes difficult to identify the semantically similar sentences and summarize it.

- **Finding the Web Forms**

  The web forms in the web pages act as a front door to the hidden web databases. So identification and mapping of the query interfaces is a major issue for crawling the hidden web content.

- **Heterogeneous schema of databases**

  A major part of the hidden web is behind the query interfaces that is stored the databases. The different databases have a different schema of tables. When the hidden web data integrator integrates the data from the different web servers, many null values and duplicate data is added in the system.

1.7. THE PROPOSED WORK

This section presents a brief discussion about the research work carried out during the course of study. The proposed technique "Dynamic Query Processing for Hidden Web Data Extraction" **(DQPHDE)** provides a medium for processing a different kind of web documents such as unstructured web documents, structured web documents and query interfaces found in WWW. Designing such a system that involves all kinds of web documents requires clustering, text mining, summarization, query interface detection and data region extraction from web documents. The proposed system provides a single platform where all kinds of information such as processed unstructured data as well as

structured data from surface web as well as from the hidden web will be made available to the user.

For the experimental purpose, the academic domain has been taken. The proposed work can be used by the researchers where they can find all kind of information such as abstracts, summary of topic, call for papers, research articles, and books etc. about the area of their interest. The system will reduce the effort of users to extract the data from both the surface web and the hidden web. DQPHDE extracts all kind of web pages from WWW and processes them. As outlined in Fig 1.4, DQPHDE has four modules namely **Clustering using Graph and Relationship Extraction (CGRE), Semantic Based Text Mining and Summarization (STMS), Web Form Extraction and Ranking (WFER) and Expert System for Data Region Extraction & Integration (ESDREI).**

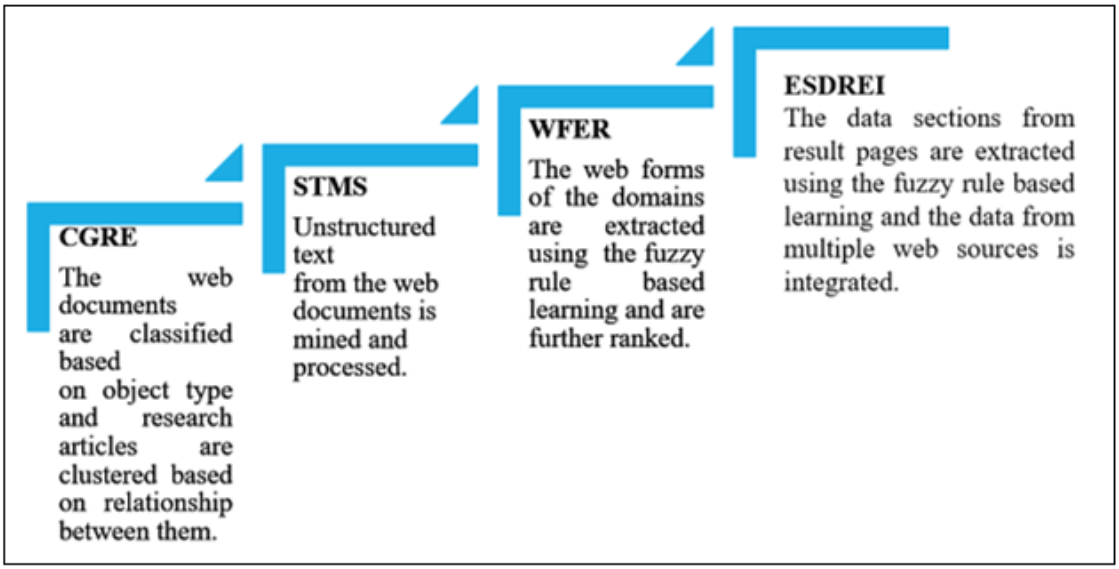

Figure 1. 4. Abstract View of DQPHDE.

1.8. STRUCTURE OF THE THESIS

The research work is principally carved into six chapters as listed below:

Chapter 1 discusses the motivation for doing the research work and discusses the concepts of hidden web. The need for extracting data from hidden web is also highlighted. It also discusses the various challenges faced in the extraction of data from hidden web.

Chapter 2 shows the working of the search engines and the different types of hidden web content. The detailed study of literature related to the proposed work has been discussed in this chapter. It explores the working and comparison with existing hidden web data extraction tools. Further, this chapter throws light on the challenges and

concludes by exploring the feasibility of deploying artificial intelligence in hidden web data extraction.

Chapter 3 provides a review of the existing techniques and algorithms used in the proposed work Dynamic Query Processing for Hidden Web Data Extraction (DQPHDE). The proposed work has used text mining, clustering, page segmentation data comparison algorithms, fuzzy rule based expert system. This chapter provides an insight into these existing techniques.

Chapter 4 furnishes four phased Dynamic Query Processing for Hidden Web Data Extraction, a novel approach which is presented in the light of drawbacks in the existing work. This chapter discusses only the first two phases of the proposed approach. The first phase constitutes by discussing clustering technique CGRE (Clustering using Graph and relationship Extraction) which uses the semantic knowledge and relationship weight for clustering the academic web documents. It further establishes relationships between web documents, creates a graph of web documents. It calculates the strength of the relationship between the web documents and clusters the web documents based on their relationship strength. The second phase Semantic Based Text Mining and Summarization (STMS) creates summary and integration of unstructured data using the semantic graph and graph operations. Phases 3 and 4 of the proposed work are being described in the next chapter.

Chapter 5 presents a technique, Web Form Extraction and Ranking (WFER) for extracting and ranking of web forms. The technique will incorporate the intelligence and knowledge owned by the expert users into the system with the aid of fuzzy based expert system. The proposed technique will automatically detect the web forms and will rank them. To aggregate the information about the entities scattered on multiple web servers fuzzy rule based technique, Expert System for Data Region Extraction and Integration (ESDREI) is being proposed.

Chapter 6 evaluates and analyses the proposed work on various measurements and claims the accuracy of the proposed work.

Chapter 7 concludes the outcome of the work. It summarizes the major achievements of the research work and elucidates the scope for future work in this domain.

1.9. CONCLUSION

"Dynamic Query Processing for Hidden Web Data Extraction" proposes to extract the data hidden behind the query interfaces using artificial intelligence and integrates

content from the hidden web as well as the surface web. The proposed system clusters the web pages using the proposed technique CGRE, summarizes and integrates the text from the unstructured web documents using the semantic based proposed technique STMS. WFER and ESDREI modules of the proposed work use the fuzzy rule based expert system to identify the query interfaces and the data regions from the web pages respectively. The proposed method is implemented using the .NET Framework and SQL. The results show that the user can get the desired result from both the surface web and the hidden web using a single query. The academic search engine prototype has been created by the proposed system. A single portal is created from which the student or the researchers can get all the required information.

CHAPTER 2

RELATED WORK IN THE AREA OF EXTRACTION OF HIDDEN WEB

2.1. HISTORY OF INTERNET

Internet was initially created to connect the isolated computers in the network, so that machines could communicate with each other. It was in 1989 that Tim Berners Lee [28], a software engineer at the CERN lab, developed set of protocols for exchange of data between computers and developed WWW. To access the information stored on different machines, uniform data structure was required. So HTML was designed to function almost identically on all computers. HTML displays data identically on all machines regardless of the hardware and the software. The documents in HTML were connected to each other by the help of the links called as hypertext. Users can click on the links and can browse the web documents. To extract information from WWW browsing and searching were the two main methods. Browsing is following the links created by the web developers. In order to browse user just has to click on the links. So browsing was easy and spontaneous. Searching on the other hand was not easy. Searching requires special software to match user keywords and extract the documents needed by the user. When the WWW was new and small, browsing was the efficient method to extract the information. But when WWW started growing, browsing became difficult as user has to manually navigate from one page to another by clicking. So search tools were developed to help the users to find the desired information easily and quickly. Two search tools were created to pull information from web. The first method was the web directory and the second was the search engine. Web directory is a context based framework for structured browsing while search engines helps in searching for a particular keyword or a phrase. Web directories work like the table of contents of a book while the search engines work like the index of a book. Web Directories use the Hierarchical Graph for storing data while search engines use Inverted Index data structures for storing the data. For example the hierarchical directory structure for storing information about the computer domain is shown in Fig 2.1. The example of Inverted Index data structure for search engines is shown in Fig 2.2.

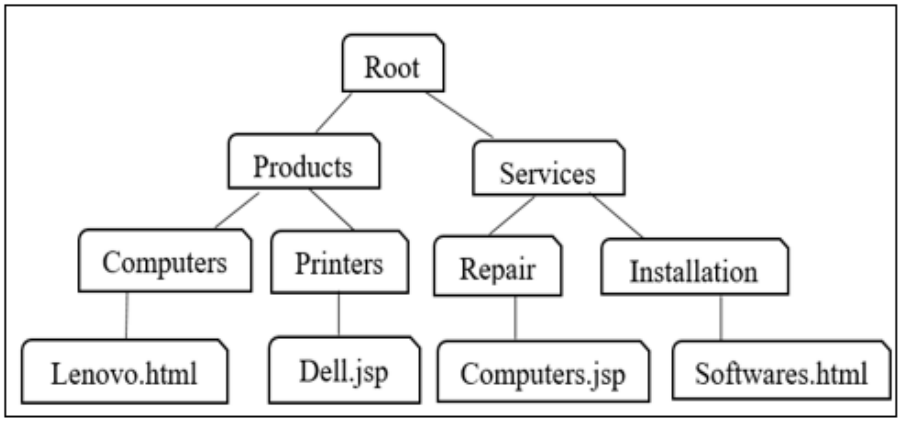

Figure 2. 1. Hierarchical Graph Structure for Web Directories.

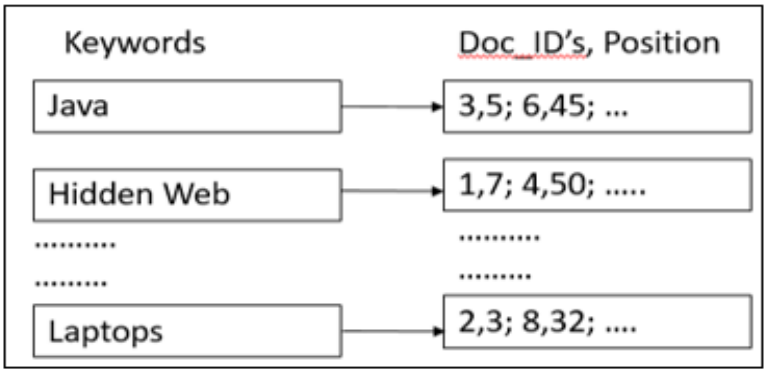

Figure 2. 2. Inverted Index Data Structure for Search Engine.

In Web Directories the top level of hierarchical graph contains the generic topic and the user has to go down the level to get to the specific topic. The web directories are not created by software but are created by humans. At the bottom of the hierarchical graph, the user is presented with the list of documents prepared by human intervention. In the case of the search engines the millions of documents are indexed in the inverted index structure. When the end user issue the query, the user keywords and the keywords in the inverted index are matched by the help of different types of algorithms. The documents which are matched and are found to be most relevant are then displayed to the users. Examples of web directories are Gopher, and Archie etc. while Google, and Yahoo etc. are examples of search engines. The differences between search engine and web directories are shown in Table 2.1.

Table 2. 1. Difference between Web Directories and Search Engines.

Web Directories	Search Engines
The web pages in web directories are organised into subject categories.	The web pages are not organised on subject categories. The special algorithm are there which rank the web pages and organise them.
Web Directories are smaller in size.	Search Engines are much larger than the web directories.
Web Directories are compiled by the humans and no software intervention is there.	Search Engines are compiled by the software and no human intervention is there.
The user query keywords are matched with the category name, site name and website annotation. The keywords are not matched with the content of the web page.	The user query keywords are matched with the content or text of the web pages.
The hierarchical graph based structure is used in the web directories.	The inverted index data structure is used for indexing in search engines

2.2. HOW THE SEARCH ENGINES WORK

A search engine is a software program which is available through internet and helps to search documents and files from WWW. The search engines have a database containing full-text index of web documents. When user issues the query on the search engines, then the user's query keywords are matched with the database and the matched results are displayed to the user. The search engines consists of three components crawler, indexer and the query processor [29]. The cyclic architecture of search engines proposed by Carlos is given in Fig 2.3.

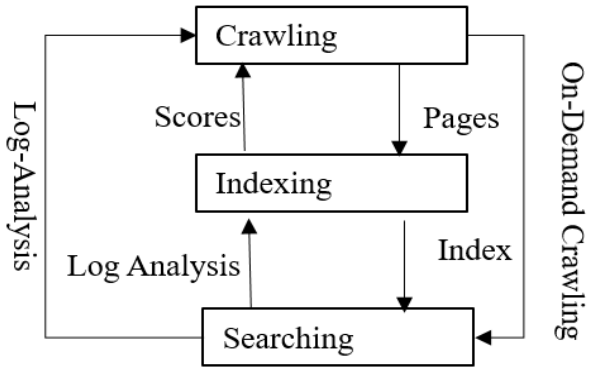

Figure 2. 3. Cyclic Architecture of Search Engine.

The first component of search engines is crawler, also called as spider, robot, search bot, which finds and fetches the web pages from WWW. The second component of search engine is indexer, which indexes words found in web pages and stores them in the database. The third module of search engine is searching, also called as query processor, which compares user query keywords with the index database and displays the relevant and matched documents.

2.2.1. Web Crawler

Web Crawler [30] [31] [32] is a software that traverses the WWW in a methodical, automated manner but in an orderly fashion. Web crawler finds and collects the web documents from WWW and then hand over these pages to the indexing module of the search engine. The web crawler starts from a set of web pages (seed URL's) and follows the hyperlinks found on web pages. The search engines may run multiple instances of web crawlers simultaneously on multiple servers. Before the crawler starts the crawling process, a list of URL's is prepared which are known as seed URL's. The web crawler start its journey from these URL's and will keep on extracting the web pages which are connected to the seed url's. These seed URL's are kept in a queue known as URL Frontier as shown in Fig 2.4. Before starting the crawling procedure the crawler also looks for a file called "robot.txt". This file is created by the website administrator, which contains instructions for the crawler. These instructions tell the crawler which part of the website is index able and which parts are non-index able and should be ignored for indexing by the crawler. All the crawlers of the search engines follow these instructions and then they start crawling the website.

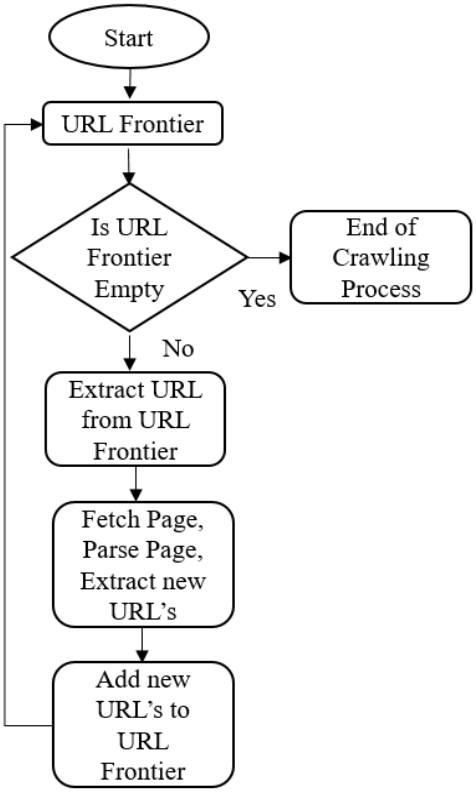

Figure 2. 4. Steps of Traditional Web Crawler.

The crawlers uses the inbuilt traversing algorithm through which it will crawl the connected set of pages. The crawlers have a discover function by the help of which it discovers new URL's by jumping from seed URL to the connecting URL and so on. The unseen or the un-crawled URL's are then placed in the URL Frontier. Sometimes there are certain web pages which are not connected to other web pages, called silos or standalone pages. These pages are difficult to be indexed and crawled. The website administrator have to put such kind of web pages in the sitemap.xml, so that the crawler can discover and crawl them. The web crawler keep on extracting the URL's from the URL Frontier until it is empty. If no URL is found in the URL Frontier than the crawling process stops else the web page from that URL will be fetched. From the fetched web page, information will extracted using different algorithms. This information extraction is known as parsing. This information extraction later helps the search engine in finding the relevant web pages as per user's query keywords. The new hyperlinks will also be extracted from the fetched web page and will be placed in the URL Frontier. This loop will go on till new unseen URL's are found.

There are different types of web crawler [33] available. Few important web crawlers are listed below.

- Simple Crawler: It was the first web crawler developed by Brian Pinkerton in 1998[34]. Simple crawler has basically four different parts such as a database, set of agents to retrieve web pages from the web, a dispatcher for the coordination of database and agents and an indexer to update the index. It was slow and had limited efficiency.
- Focused Web Crawler: A focused web crawler collects the web pages on specific set of topics. The concept of focused web crawler was introduced by Manczer in 1998. They was also called as topical crawlers. The few examples of focused web crawlers are Healthline, Codase [35].
- Parallel Web Crawler: Parallel web crawler was introduced by Junghoo Choo in 2002 [36].A Parallel Web Crawler has multiple crawling processes. Each crawling process has the same functionality as that of a single-process crawler. Each crawler downloads the web pages from WWW, stores them in the database, extracts URL's from them and follows the extracted hyperlinks. If one crawler process has many URL's to explore then the load of crawling from that process can be assigned to lightly load crawling process. For example, Google crawler has parallel, single-threaded processes [36] [37].
- Mobile Web Crawlers: The concept of mobile crawlers was introduced by J.Hammer and J. Fiedler [38]. The crawlers here can migrate to the actual data source i.e. the web servers before the actual crawling has started on that web data source. So these types of crawlers have the advantage of local data access. The data form the web servers was then stored in the web crawler's memory and then the crawler move to the next web server.
- Distributed Web Crawlers: The distributed web crawlers were introduced by Paolo Boldi [39]. In distributed web crawlers multiple computers were used for the crawling purpose. The main aim of the distributed web crawlers was to reduce the load of crawling from a single computer and divide load among multiple computers.
- Hidden Web Crawler

  Like the traditional web crawlers, hidden web crawlers take seed URL's, download pages, process the pages and extract the links from the downloaded pages. Unlike the traditional web crawlers, the hidden web crawlers were able to distinguish between the web pages containing the web forms and the web

pages which don't have them. The hidden web crawlers performed several actions on web pages having forms such as form detection, form analysis, value assignment, form submission and response analysis.

2.2.2. Indexer

When a user issues his/her query on the search engine then web pages are not processed again. Instead the search engine builds a database of the keywords found in crawled web pages. The main aim of the indexer module is to find the information as quickly as possible. Without indexing information searching from the search engines have been a very time consuming and difficult task. The web pages crawled by the crawler acts as an input to the indexer module of the search engine. For indexing the indexer looks at the metatag and the keywords found in the web pages. The meta tags helped the owner of the web page to specify the domain or concept of the page. There are three steps of the indexing module namely text acquisition, text transformation and index creation as shown in Fig 2.5.

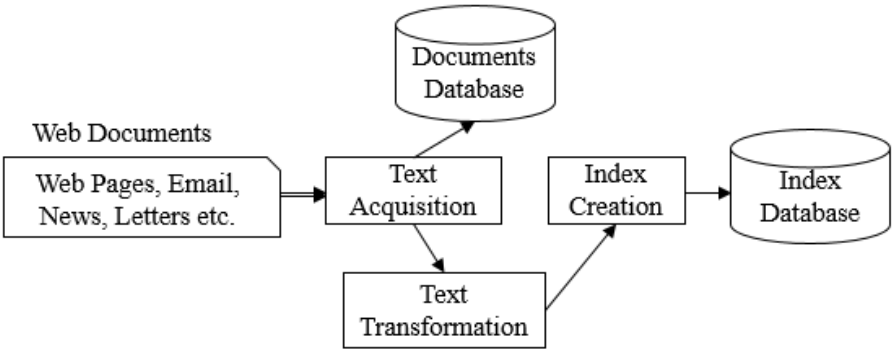

Figure 2. 5. The Indexing Process of Search Engine.

- Text Acquisition

Text acquisition component of indexer identifies and extracts the web pages. The text acquisition performs many functions. The first function is to download the web documents. The second function is document feeds, which is a mechanism for accessing the bulky data which changes frequently in real time. The third function is document conversion. The web documents downloaded can be in different formats such as HTML, XML, PDF etc. The conversion component converts the data from the files in different formats into text and metadata format. The last function of text acquisition is creating the document data store. The documents are stored in a compressed form in order to improve the efficiency of the search engine.

- Text Transformation

The task of text transformation is to convert the text from the document into index terms or features. When the end user issues his/her query, the index term are used for searching documents. The different functions of task transformation are parsing, stop word removal, stemming, link analysis, information extraction and classifier.

- Index Creation

The output of text transformation acts as an input to index creation component. The main aim of index creation is the creation of the index and building the data structures that will help in fast searching of the web pages. The functions of index creation are document statistical analysis, weighting, inversion and index distribution. In document statistical analysis the statistical information about words, features and documents is gathered and recorded. This information is further used for ranking the web documents. The weighing functionality computes the relative importance of words in the document, which is further used for computing scores of keywords and ranking of web pages. The term frequency and inverse document frequency is computed by weighing. The inversion functionality is the core function of the index creation module. Its task is to convert the document-term information into term-document information in order to create the inverted indexes. The index distribution function distribute the index created to multiple computers. The index distribution can later help in the parallel processing of the queries.

2.2.3. Query Processor

The query processor compares the user query terms with the index terms and returns the most relevant documents to the user. The query processor has three main steps namely user interaction, ranking and evaluation as shown in Fig 2.6.

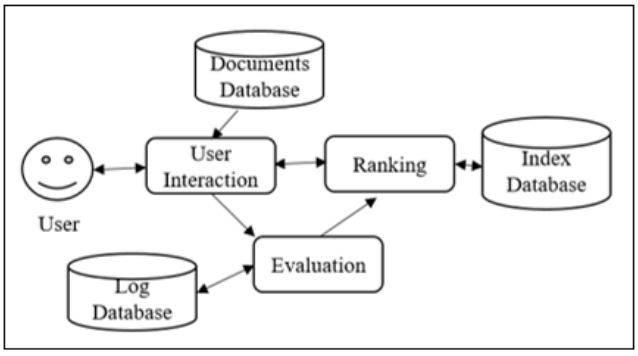

Figure 2. 6. The Query Processor Module of Search Engine.

User Interaction

The user interaction component acts as an interface between the end user and the search engine. The main tasks of user interaction components are given below.

- Accepting user's query
- Conversion of user query into index terms or tokens
- Extract the ranked list of web documents from the search engine and organize them and then show them to the end users.
- Provides the techniques to the user to refine their input query.

Ranking

This component is the core component of the search engines. It takes the processed user query as an input which is generated by the user interaction component and generates a ranked list of web documents as an output. Ranking technique must be efficient and effective because it is the ranking module which determines whether the search engine achieves its aim of fetching and finding relevant documents or not.

Evaluation

The main aim of evaluation is to measure the effectiveness and efficiency of the search engine. The user behaviour while browsing is recorded in the log and it is further analysed offline. The results of the evaluation component are used to improve the performance of the ranking component.

2.3. THE DIFFERENT COMPONENTS OF WWW

World Wide Web is a collection of web pages created using HTML and accessed using the HTTP protocol. The World Wide Web was developed in 1991 by Tim Berners-Lee [28]. World Wide Web is also known as "Web". The three components of WWW are shown in Fig 2.7.

- Surface Web
- Hidden Web
- Dark Web

Figure 2. 7. Components of WWW.

2.3.1. Surface Web

The surface web is also known as indexable web, visible web, lightnet. It is the part of WWW which is easily accessible by users and is indexable as well as stored by search engines as shown in Fig 2.8.

Figure 2. 8. Surface Web.

By the help of web crawlers the search engines create web page repositories. The web crawlers parse the web pages, extract terms and pass this information to the indexer. The indexer stores the keywords and the location of the keywords found in the web pages. All the web pages which are connected to each other by the hyperlinks are downloaded and stored by the search engines. Such kind of pages which are easily reachable by the crawlers and are indexed are known as the surface web. According to the study conducted by “www.worldwidewebsize.com” as on 6 August 2018, Google index 52 million web documents and Bing indexes 0.16 billion web pages as shown in Fig 2.9 and Fig 2.10.

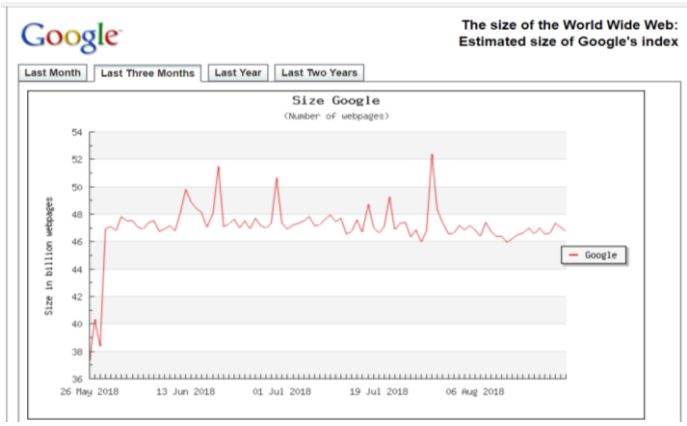

Figure 2. 9. The size of the World Wide Web: Estimated size of Google's index.

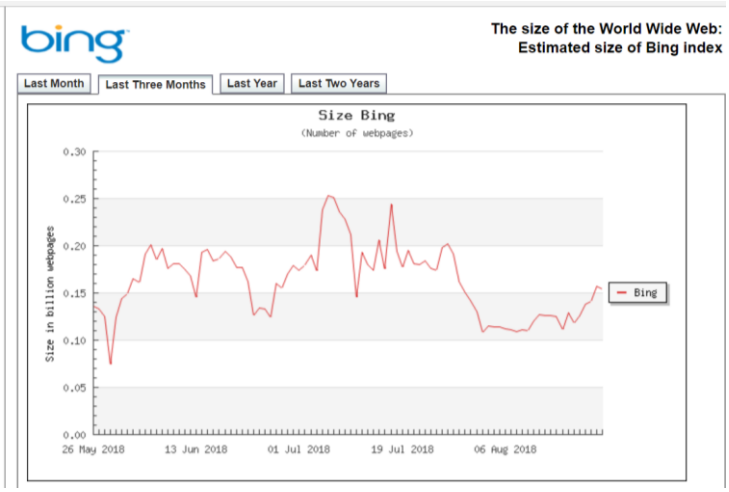

Figure 2. 10. The size of the World Wide Web: Estimated size Bing's index.

2.3.2. Hidden Web

The hidden web is also known as the invisible web, deep web or the non-indexable web. It is the part of WWW which is not crawled and indexed by the traditional search engines as shown in Fig 2.11 [122]. The hidden web is mostly comprised of the web database, unlinked web pages, authentication requiring web pages and the dynamically generated web pages.

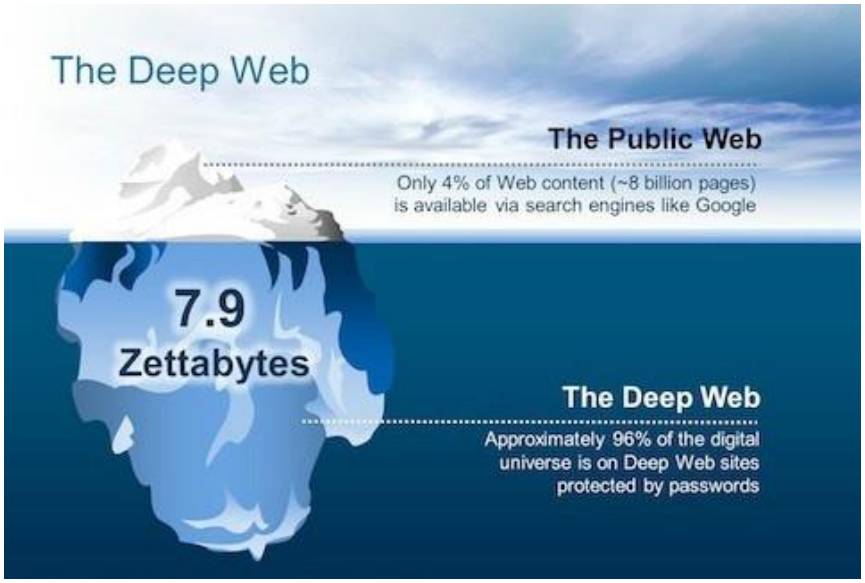

Figure 2. 11. Hidden Web.

Such kind of pages are not indexed by Google or Bing. The major part of hidden web is accessible through the query interfaces or web forms. The end user has to fill the web forms, submit the query and only then they are able to access the data behind the query web forms. This data is not hidden from the end users but it is hidden from the search engines. The hidden web is high in both quantity and quality. According to the study conducted by Bergman major part of the hidden web resides in the databases, which is accessible by the help of the web forms. The web forms or the query interfaces acts as a doorway to access the hidden web. To access this hidden data end users fill the web forms, submit the forms and extract the data from the hidden web databases as shown in Fig. 2.12.

Figure 2. 12. User Interaction with Web Forms.

Types of Invisibility in Hidden Web

The traditional web crawlers and search engines are not able to process the web forms automatically, so the hidden web remains unreachable from the search engines. There are various reasons of invisibility of the hidden web. Depending on the reasons of the invisibility the hidden web is of four types.

1) Opaque Web
2) Secret Web
3) Proper Invisible Web
4) Proprietary Web

- **Opaque Web**

These kind of web pages are invisible to the search engines because of some technical issues. If these technical issues can be resolved, then these web pages can be indexed by the search engines. The first technical issue is the depth of the crawl. The web crawling process is a very resource consuming process. So the web crawlers do not crawl each and every web pages of a website. They fix their depth of the crawling in order to optimize the crawling process resource utilization [121] as shown in Fig 2.13. Because of this reason few web pages remain hidden as they are not indexed.

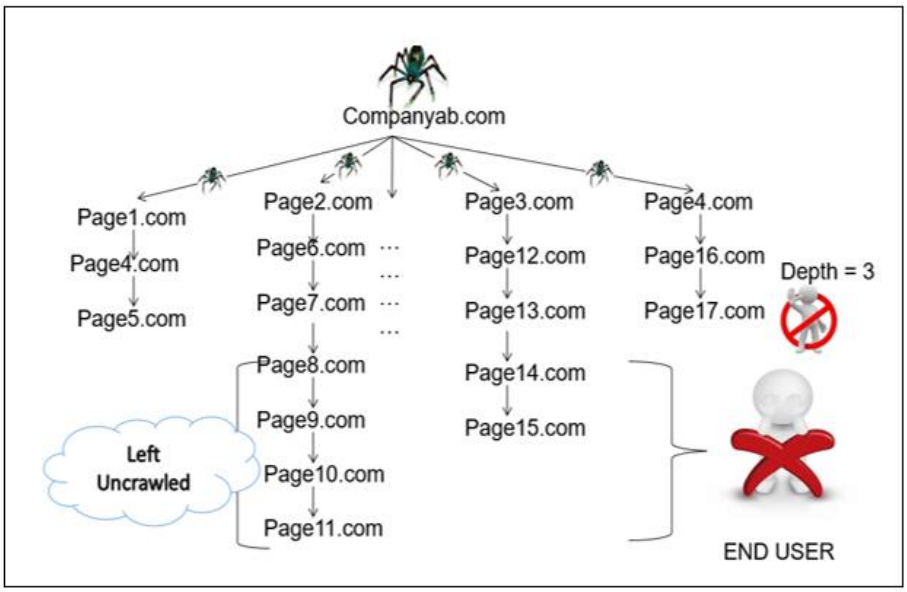

Figure 2. 13. Opaque Web: Depth of Crawl.

The second technical reason because of which web pages are not indexed by the search engines is the disconnected URL's. Certain times few web pages of a website are not linked to other web pages of that website, neither are they linked to web pages of the

other web sites. So they remain undiscovered by the web crawlers and hence are not indexed by the search engines as shown in Fig 2.14.

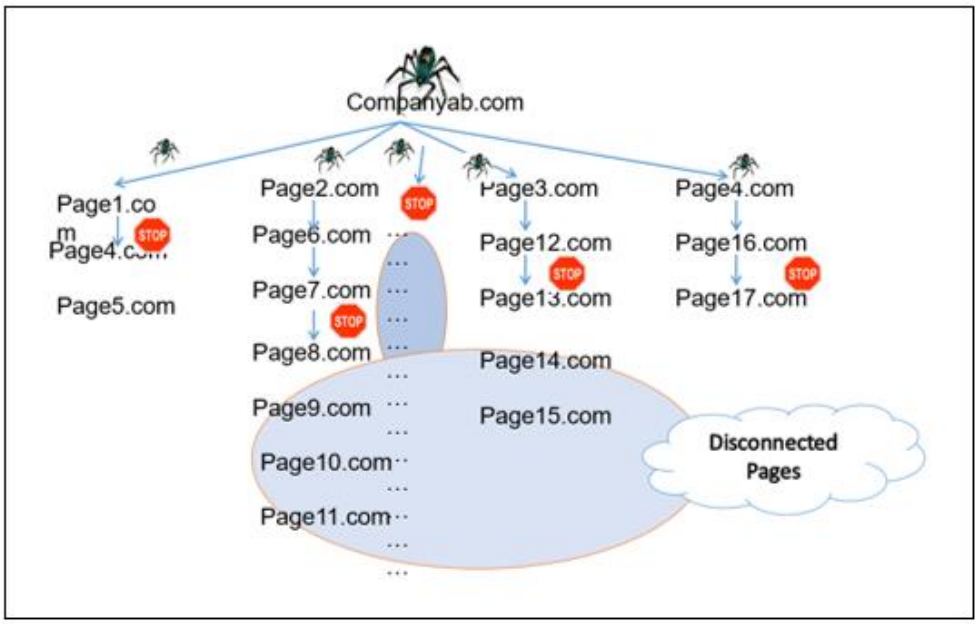

Figure 2. 14. Opaque Web: Disconnected URL's.

The third technical reason behind the invisibility in the opaque web is the frequency of crawling. The web crawlers crawl the WWW after a certain time interval. Meanwhile thousands of web pages are deleted, updated and many more new web pages are added into the WWW as shown in Fig 2.15. So the newly added web pages remain hidden from the search engine until its web crawler, crawls again.

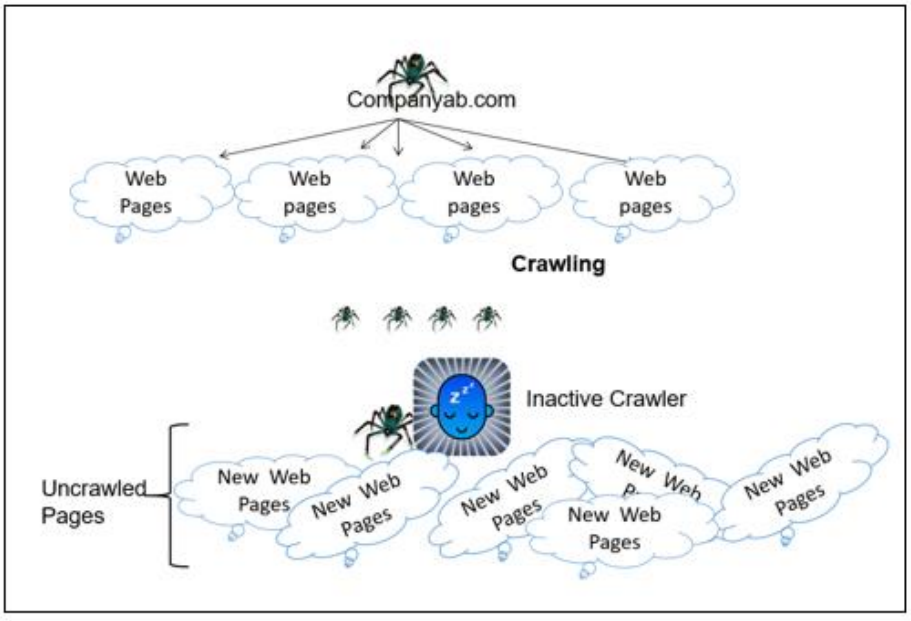

Figure 2. 15. Opaque Web: Frequency of Crawling.

- **Secret Web**

These web pages are hidden because the website administrator does not wish these web pages to be seen by the crawler and indexed by the search engine. There is no technical reason because of which these web pages are hidden. There are several techniques by the help of which the website administrator can prohibit the web crawlers to crawl the web pages. In the first technique robot exclusion protocol is used. The Website administrator make a list of web pages which should not be crawled and places that list in the robot.txt file. The web crawler obeys the rules and before starting crawling a particular website, the crawler reads the robot.txt file and simply ignores the web pages listed there as shown in Fig 2.16.

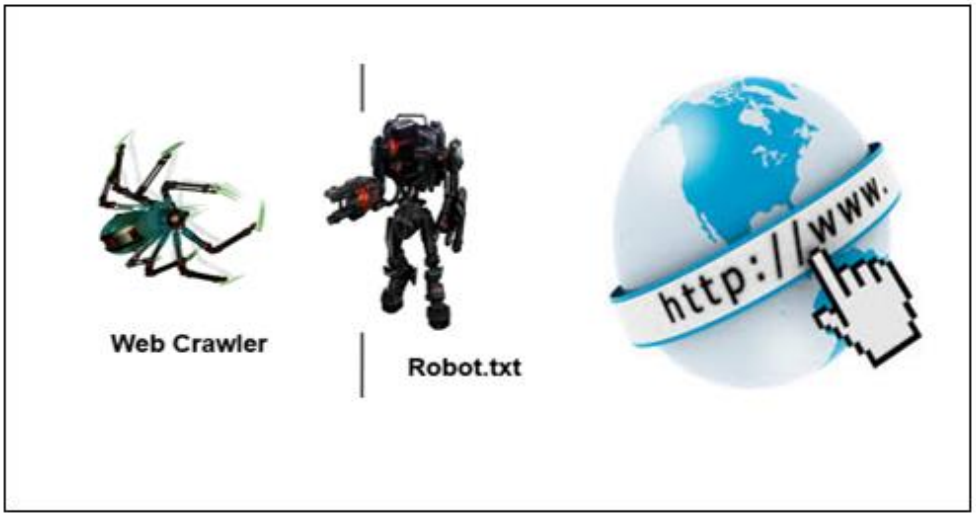

Figure 2. 16. Secret Web: Robot Exclusion Protocol.

The second technique make use of no-index protocol. The webpage developer places the no-index Meta tag in the web pages which should not be indexed. The no-index meta tag is the page level policy while robot.txt is the website level policy.

The web pages are made accessible to only authorized web users by the third technique. Certain web pages are accessible only when login forms are processed. As the search engines are not able to provide the user id and password in the login form, so the web pages behind the login forms remain hidden to the web crawlers. Not every user is allowed to view those pages, only the authorized users are allowed to access them as shown in Fig. 2.17

Figure 2. 17. Secret Web: Password Protected.

- **Proper Invisible Web**

There are technical reasons because of which the pages in the Proper Invisible Web are hidden. There are two reasons for invisibility the query interfaces and format of the files. There are certain web pages which are accessible through the query interfaces only. Unlike the end users, the web crawlers are not able to process the web forms and extract the web pages behind the query interfaces. The second reason of invisibility is the different format of files found in WWW. The file formats such as pdf, doc, ppt, audio, and video etc. are found on WWW. The web crawlers don not process these file formats sometimes because of technical reasons and certain times because of the spider trap problems.

- **Proprietary Invisible Web**

There are web pages in the hidden web which are only visible to those set of people who agrees to certain conditions before viewing the web pages. The registration procedure is required where users enter their personal details and register themselves. As the web crawlers are not able to complete the registration procedure, so they fail to crawl these set of pages too.

2.3.3. Dark Web

It is the part of WWW which is not indexed by the traditional search engines. The data of the dark web is not intended to be seen by the users. The data of the dark web is

encrypted, sometimes because of some legitimate reasons and certain times to hide the illegal activities as shown in Fig 2.18.

Figure 2. 18. Dark Web.

The normal web browser are not enough to access this part of WWW. Special web browser such as TOR etc. are designed to access dark web. The TOR web browser hide the activities of the users and make the users identity untraceable.

2.4. HISTORY OF HIDDEN WEB

In the beginning of internet development there were few websites and web pages. So the web pages were easily crawled by the search engines. Later three technologies came on the surface such as database technologies, e-commerce and the dynamic web pages. By the help of these the large websites using database started developing. These web sites no longer have static web pages. The e-commerce companies started creating an entirely new class of website. This class of websites have dynamic web pages which fetch data from the database servers. The traditional search engines were not able to index the content of this new class of websites.

Dr. Jill Ellsworth in 1994, coined the term invisible web, which refers to the content not indexed by the search engines. According to the study conducted by Bergman in 2004, hidden or invisible web was found to be 500 times larger than the surface web. Bright Planet [25] performed the study and analysis on hidden web. Their key findings include:

1. Hidden Web is 400 to 550 times larger than the surface web
2. The hidden web contains 7500 TB of information, while surface web has 19 TB of data.

3. Hidden Web has 550 billion web documents while surface web has one billion documents.
4. The hidden web has high quality of data and is more relevant.
5. Comprehensive

2.5. QUERY INTERFACE EXTRACTORS

In today's world people from all over the world depends on the internet for accessing the information. They search the web pages from WWW according to their requirement by issuing the queries on the search engines. But the search engines are able to index only a part of WWW, which is surface web. A major part of WWW, hidden web, is not indexed by traditional search engines. The WWW contains a humongous amount of data in the database servers which is only accessible through the query interfaces and the search engines are unable to index them [40]. The data behind these query interfaces forms the hidden web and according to study conducted by Bergman, hidden web is 500 times the size of the surface web [25]. The hidden web covers a great variety of subject areas, ranging from business, government, education, to entertainment [41, 25]. There are thousands of databases for every domain on the web. The data in these databases is of very high quality and quantity. In order to fetch this valuable and huge amount of data, users interact with multiple query interfaces and issue their queries to the databases. It consumes a lot of effort and time of the end users. So the problem of fetching the data from the hidden web has got a lot of attention from both the researchers and the industries. In the beginning the work includes [42-46] and later efforts include [47-55]. To fetch the data behind the query interfaces a lot of work has been done for understanding the query interfaces and merging them to create a unified query interface. The tools and the techniques developed for extracting and creating the unified query interfaces have been discussed below.

A generalized query interface using the semantic relationships has been created by Deepak [56]. In the technique an algorithm has been developed for understanding the model of different query interfaces of a particular domain and finally integrating them to produce a generalized query interface. The integration of query interfaces has been done in two steps. In the first step the schema matching has been done which extracts the semantic similarities between the attributes found in the query interface. In the second step the schema integration has been done which creates a unified query

interface using the output of semantic matching. These two steps have been discussed below.

Step 1: Semantic Matching

In this step the estimated similarity values (ESV) between the attributes of different query interfaces has been calculated and stored in Similarity Value Matrix (SVM). The threshold value of similarity value has been set. The ESV's or mappings which were below the threshold were discarded, rest were retained and stored in Mapping Knowledge Base as shown in Fig 2.19.

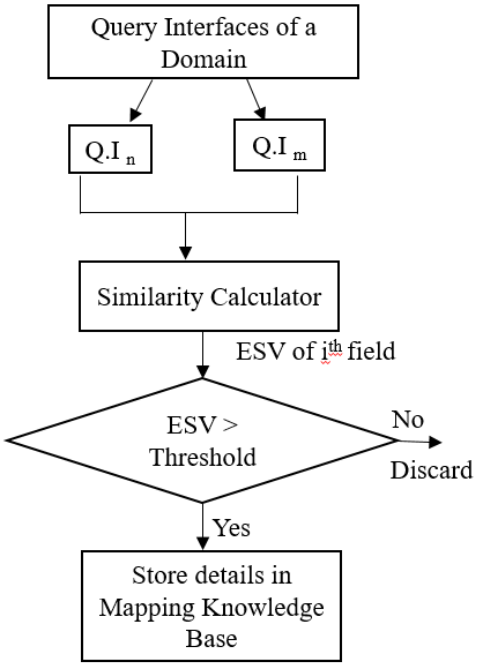

Figure 2. 19. Semantic Matching for Creating Generalized Query Interface.

The mappings in the Mapping Knowledge Base have been stored as clusters. Within a given domain, a cluster contains all the attributes of different query interfaces that were semantically equivalent. The Mapping knowledge Base has four fields namely Cluster Id, Cluster Name, Schema Name, and Field Name.

Step 2: Schema Integration

In this step the integration of query interfaces have been done pairwise i.e. initially q1 and q2 were integrated as q12 then q12 was integrated with q3 and so on. The integration algorithm has used the following functions for merging the different query interfaces of the same domain.

- Relation (e, f): This function finds the semantic relationship between attributes e and f. If the similarity is there it returns 1 else it returns 0.
- Combinesubtrees (e, f, ef): This function combines the two query interfaces e and f, which are represented as tree, and creates a merged subtree "ef".
- Assignname (ef): This function gives the name to the merged query interface.
- Removedupnodes (ef): This function removes the duplicate nodes from the subtree "ef".
- Insert (ef, Sij): It inserts the sub query interface "ef" in the merged query interface Sij

Another new approach VIQI, has been proposed by Radhouane Boughammoura[57], which interpreted the query interfaces as done by the users and extracts query from query interfaces. In VIQI a new technique has been developed which includes models for query representation and query interpretation and extraction. These models have been discussed below.

Model for Query Representation: The query has been represented using the hierarchical model by VIQI. Based on spatial-locality paradigm, the form fields which are of same concept are located close to each other in the query interface while of different concepts are rarely adjacent and aligned. So in VIQI on the basis of the mapping, using the visual clues and the semantic similarity of concepts in the interface, five components have been identified in its model. These components are

- Field
- Rendered Group
- Rendered Collection
- NonRenderedGroup
- Visual Box

Model of Query Interpretation and Extraction: In VIQI the query interfaces have been interpreted as done by the users. As the user is able to identify the similar concepts in a query interface by the help of the proximity and alignment between fields, similarly VIQI used this to interpret the query interfaces. The Euclidean distance has been calculated to measure the closeness between the attributes. Then the alignment between fields is calculated using the AlignX (f1, f2) function. X can have four values Buttom, Top, Left and Right. Then using the alignment and Euclidean distance the proximity

has been calculated using equation 1. The DBSCAN (Density Based Spatial Clustering of Applications with Noise [58]) has been used to group fields.

$$\text{Proximity} = \frac{\text{DistEucl}(f1, f2)}{\text{Align}(f1, f2)} \tag{1}$$

WeiFeng Su have introduced another technique called Statistical Parser (StatParser) [59], which harnesses the power of both the rule based methods and learning based methods to understand the query capabilities of search forms. First of all learning has been done from a set of parsed query interfaces and then it parses the new query interfaces into a hierarchical representation. The StatParser has two phases as shown in Fig 2.20.

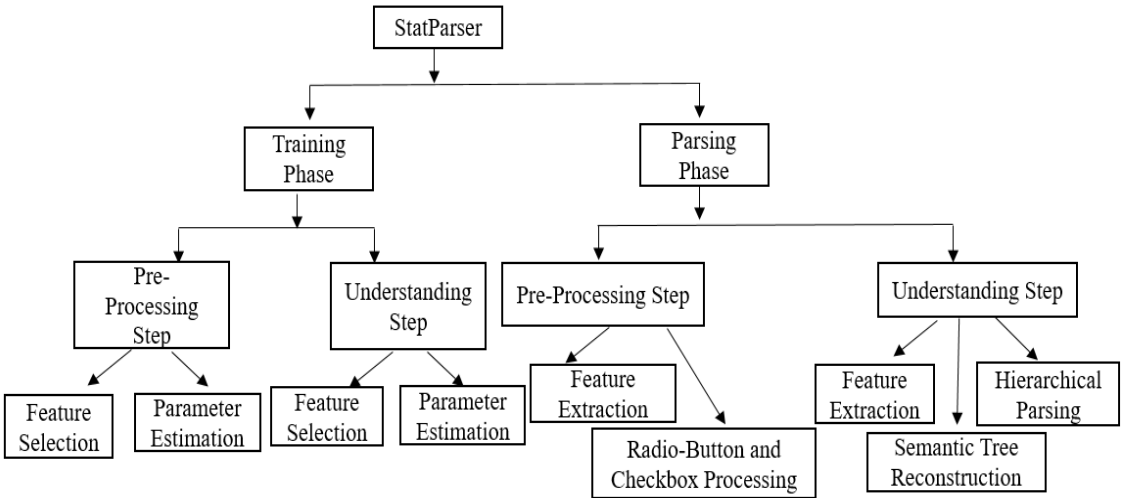

Figure 2. 20. Different Phases of Stat Parser.

- Training Phase
- Parsing Phase

Training Phase: In the training phase the important features and their respective parameters have been estimated with a set of parsed query interfaces. It has further two phases the Pre-processing Phase and the Understanding Phase. In the Pre-Processing Phase the radio-buttons and the checkboxes that share the same name value have been combined and their labels have been combined into a single element. The pre-processing phase of training has two steps the Feature Selection and the Parameter Estimation. Feature Selection selects a set of features E_p, from a given set of query interfaces, which were found important for the processing of radio-button/checkbox

processing. Parameter Estimation learns the probability parameters of the selected features. The Understanding Phase generated the parse tree from the pre-processed query interfaces, I_p. The understanding phase has two components Feature Selection and Parameter Estimation. These two components of understanding phase have the same functions as those in the pre-processing phase except that they use different grammar for different purposes.

Parsing Phase: The features and the parameters extracted in the training phase have been used here to understand the new query interfaces. Parsing phase too has two phases namely Pre-processing and the Understanding phase. The pre-processing phase has two steps namely Feature Extraction and the Radio-button and Checkbox processing. The feature extraction extracted all the features E_p from the query interfaces for each group of radio-buttons/checkboxes and labels around it. In the radio-button and checkbox processing the radio-buttons/checkboxes and labels have been combined according to the identified features and learned features under the maximum entropy principle. The understanding phase of parsing has three steps namely Feature Extraction, Hierarchical Parsing and the Semantic Tree Reconstruction. Feature Extraction extracted all the features from the query interface "I" for each element/label, text edit box, selection list and simplified radio-button/checkbox group. In the Hierarchical Parsing step, the pre-processed query interfaces have been parsed into a parse tree having maximum probabilistic score according to the identified features and learned parameters. In the Semantic Tree Reconstruction the parse tree has been converted into a semantic tree.

Tian Zhao also proposed a new adaptive and optimised RDF query interface [60]. By the help of this RDF query interface, the user can query and synthesize a distributed Web Feature Service (WFS) data as shown in Fig 2.21.

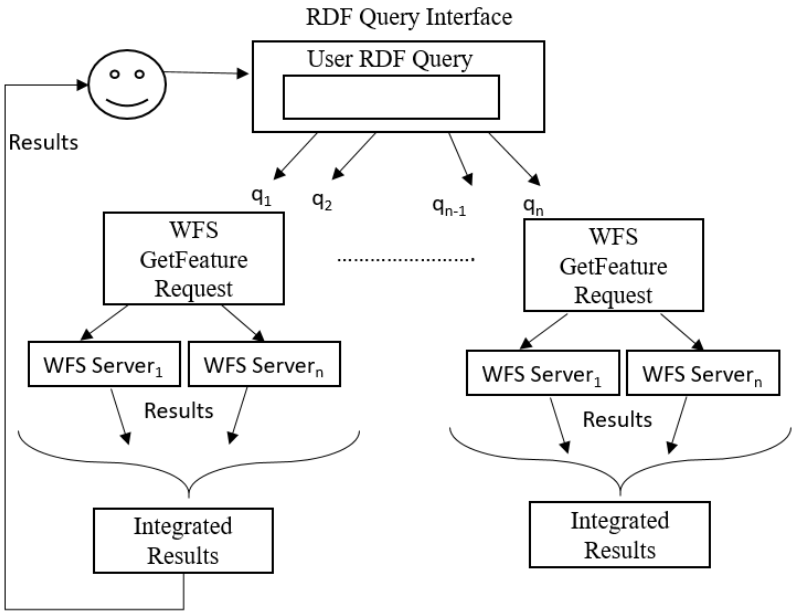

Figure 2. 21. Architecture to Create an Adaptive and Optimized Query Interface.

Each query written by user in SPARQL-like syntax has been translate to WFS GetFeature requests. These requests were then sent asynchronously and simultaneously to WFS servers. The multiple responses from WFS servers have been integrated to answer the user's initial query.

Wensheng Wu has proposed a new technique ExQ [61], which is an interface modelling and extraction system. ExQ first of all extracted the structure of the query interfaces using the visual representation via spatial clustering and then marked the schema with the labels as done by the human. ExQ took the query interface as an input and produced the schema of the interface as an output. ExQ has two major components structure extractor and schema annotator as shown in Fig 2.22.

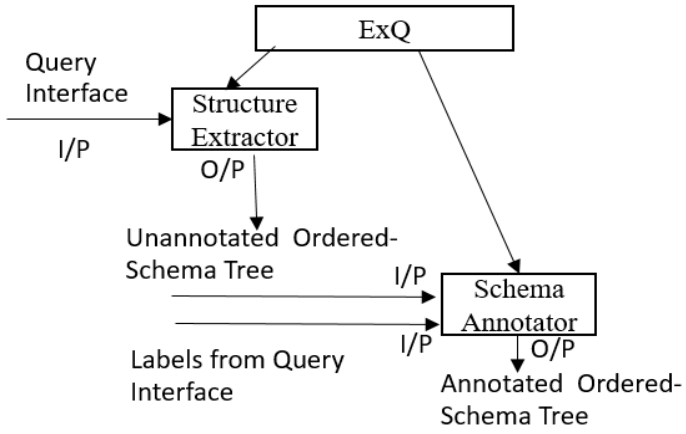

Figure 2. 22. The Working of ExQ.

Structure Extractor: For extracting the structure hierarchical agglomerative clustering algorithm has been used. The algorithm used the distance, alignment and the direction of the attributes of the query interface to cluster them and find the relationship between them. It took the query interface as an input and produced the unannotated ordered-tree schema of the interface as an output. Initially there were a set of clusters, each cluster contained the single attribute. Then the clusters have been merged repeatedly until there was a single cluster containing all the attributes. For merging the attributes or clusters the information such as distance among the attributes, section titles, and horizontal lines have been used. To look for these information, the grouping patterns have been used. The grouping pattern specifies the way to group the group the attributes of the query interface. The grouping patterns can be separator-based patterns, alignment-based patterns or indentation-based patterns.

Schema Annotator: This step finds the labels from the query interface for the nodes of the ordered-tree schema generated in above step. For annotating the schema, an annotation block has been defined which is the smallest rectangular block containing the label and the attribute. The label attachment algorithm ATTACH accepted the unannotated ordered-tree schema of a query interface, set all the labels for that query interface and returned an annotated schema tree as an output. ATTACH annotated the nodes in the schema tree in group in the post-order. The child nodes in the group have been annotated by ATTACHONE algorithm. ATTACHONE takes input group "N" and set of labels L_a as an input. Some labels from L_a have been assigned to the nodes from the input group "N" and left labels have been returned. ATTACHONE algorithm has three major steps candidate generation, candidate pruning and match selection.

2.6. HIDDEN WEB DATA EXTRACTORS AND INTEGRATORS

Hidden Data Integrators is an approach of accessing the hidden web content where the data of a domain from multiple sites is integrated. The aim of integration is achieved by creating a mediator system, one for each type of domain (e.g., books, cars, hotels etc.). The mediator schema for each domain is created automatically by analysing forms of that domain [72, 73]. The input form is then analysed and the domain of the form is identified. The mapping between the input form and the mediator schema is done. The queries over the mediated form are then translated into the input form queries. The queries are then fired on multiple web forms. The results are downloaded, processed,

integrated, ranked and then shown to the users. The work done by the researchers to integrate the hidden web data is discussed below.

A system Dexter [74] has been developed which extracts the websites having product details and extracts the products detail from them. A big collection of product specifications has been built by the system Dexter. The collection consisted of specifications from instances of a given product from each visited website. It employed the scalable focused-crawling technique to obtain specifications in a domain of interest. Dexter started from few seed URL's and then used the vote-filter iterate principle. In each iteration it used vote and filtering technique to obtain large set of product description, remove irrelevant websites or web pages in a website and reduced the noise. There were four steps in the architecture of Dexter.

- Website Discovery
  It extracted the websites which contains the specification of the products. Ranking of the extracted websites was done for further exploration.
- In-Site Crawling
  A website has multiple web pages. From those webpages the pages which contained the product specifications were extracted.
- Specification Detection
  From the web pages containing the product specifications, the HTML fragments were detected which contained the product specification.
- Specification Extraction
  The last step in Dexter was to extract the attribute-value pairs from the extracted HTML fragments.

Deepbot [75] was a hidden web crawler developed by Manuel Alvarez. The hidden web content from both client side as well as server side has been extracted by Deepbot. For handling the client side data client side scripting is handled and session management is done. The other conventional crawlers were not able to handle client-side scripts and sessions management. For handling the client side scripts Deepbot has a mini web browser. The details of the domain have been stored in Deepbot for crawling server side hidden data. These details helped in fetching data behind query interfaces which traditional crawlers failed to do. Deepbot has seven components and these components are:

- Route Manager Component

  Route Manager created and managed the set of URL's that would be crawled. This set of URL's is called the master route list. For crawling Deepbot has a pool of crawlers and the master route list has been made accessible to all crawlers from the pool. Route manager also handled the session management.

- Configuration Manager Component

  Configuration manager stored the start-up information for the crawlers such as information about the seed URL's, the depth of crawl, different download handlers for downloading different kind of files (images, Pdf, MS Word, html etc.), selecting the important URL's etc.

- Download Manager Component

  The mini web browser has been used for crawling and downloading the web documents. The domain definitions stored by Deepbot has been used by the mini web browser for crawling purpose. Download Manager Components selects the appropriate handler depending the type of download document (images, Pdf, MS Word, html etc.)

- Content Manager Component

  Content manager analysed the downloaded web documents using filters. Three filters have been used for analysing the downloaded web pages. By the help of first filter the relevancy and quality check-up has been done. If quality of web document has been found poor then that web pages has been discarded else web page has been stored. The other filter known as obtain links extracted the URL's from the selected downloaded web pages. The last filter called form analyser filter analyses every form and determines the relevancy for any of the preconfigured domain definitions.

- Data Repository

  The database of web pages downloaded from both surface web and hidden web has been maintained by data repository.

- Indexer

  The keywords from the downloaded web documents have been extracted and get stored by indexer. It helped in fast access of web document when users fired the queries.

- Searcher

  The searcher processed the user query. The generic searcher and the ActiveX searcher have been used to fetch the documents of user interest.

A task-specific hidden web crawler Hidden Web Exposer (HiWE) was developed by Sriram[76]. HiWE was developed to extract data behind the web forms. HiWE processed, analysed and submitted web forms. HiWE has seven components namely, URL list, crawl manager, parser, form analyser, form processor, response analyser and label value set.

- URL List

  The URL's that have to be crawled get stored in the URL list.
- Crawl Manager

  The crawler manager helped in the crawler process. The websites that need to be crawled have been provided as input to crawl manager.
- Parser

  The parser has the responsibility of reading the downloaded web documents and extracting the URL's from pages. These extracted URL's then get stored in URL list.
- Form Analyser

  The main task of form analyser was reading web form and extracting information from it. Layout Based Information Extraction technique has been used by form analyser for extracting the semantic information from the web forms. Form analyser stored the extracted information of web form as shown below.

  $$(F= (\{E1, E2…En\}, S, M)).$$

  Here E1, E2… En were the form elements, S was the submission information and M was the meta tag information of the web pages.
- Form Processor

  The data behind web form has been extracted automatically by filling the web query interfaces and submitting them. The LVS table has been used by form processor to automatically fill the web forms.

- Response Analyser

  After the form processing has been done by form processor the result pages were returned. These result pages were analysed by response analyser. The result pages get stored in the crawler's repository. The web pages have been categorised as result pages and error message containing web pages. This analysis of result pages helps in later assignment of values to the form elements.
- Label Value Set(LVS)

  The web forms have been automatically filled by HiWE by using values stored in LVS table. LVS table has maintained label and value set. The value set has a value and a number that indicates the effectiveness of that value.

An incremental crawler has been proposed by Dr. Komal Kumar Bhatia [77]. The main aim of crawler was to maintain the freshness of database containing web pages. The aim has been achieved by re-crawling the web pages that were updated more frequently. Based on the probability of updation of web pages, the time period between two successive revist of crawler has been set. The incremental web crawler has the following components:

- Domain Specific Hidden web crawler (DSHWC)

  From eth downloaded web forms unified query interface has been created by DSHWC. The unified query interfaces has been filled and submitted automatically. The result pages have been downloaded and stored in hidden web pages database. The URL's of the result pages also get stored.
- URL Extractor

  The aim of URL extractor was to parse the web pages from hidden web pages database and extract the URL's. The extracted URL's were then stored in AllURL data structure.
- AllURL

  AllURL data structure stored the URL's as well as other information of all downloaded web pages. The information stored helped in calculating the revisit frequency.
- Revisit Frequency Calculator

  Incremental Web Crawler was created to maintain the freshness of hidden web pages database. To achieve the objective revisit frequency calculator was used. If a web page changed frequently then that web page was visited frequently.

The frequency of change in a web page was limited to a threshold value. After this limit or threshold value (T) the revisit frequency remain constant.

- The Update Module

  The inverse of revisit frequency (τ) has been computed by the update module. If a URL has been fetched and its τ is more than T, then that particular web page was re-crawled else there was no need of recrawling and URL was not stored in URL buffer. In the end URL buffer contained the list of all URL's the needed to be re-crawled. The signal "Fetch URL" was sent to the Dispatcher module by the update module when the URL buffer has been set as full.

- Dispatcher

  The "Fetch URL" signal was sent to dispatcher and after that dispatcher fetched the URL from the URL buffer. DSHWC then re-download these web pages from these URL's.

2.7. ACADEMIC SEARCH ENGINES

Academic search engines (ASE) are those search engines which fetch the content relevant to the academics and that content is not searchable by general purpose traditional search engines. Indexing this content has been the main aim of the ASE. The academic hidden web consisted of databases and records relevant to the academics. The advanced academic search tools have been available these days but still users have to rely on the general purpose search engines. The features of general search engines, such as searching capabilities, user friendliness, simplicity, search velocity and broad coverage, help the users in their general search. But when the users look for academic contents then these features are not available to the users. The academic content contains data from both the surface web and the hidden web. The ASE should also incorporate the features of general search engines in order to ease the task of academic users.

The list of popular academic search engine available are given below.

1. Directory of Open Access Journals (DOAJ)
2. Computing Research and Education (CORE)
3. Bielefeld Academic Search Engine (BASE)
4. Google Scholar
5. Microsoft Academic

6. Research Gate
7. Academia.edu

The limitations of these search engines is lack of integration of all kinds of information, such as call for papers of the desired area, tutorial or introduction of the area, abstracts of the recent research papers of the area, books etc., which are needed by the researchers.

2.8. HIDDEN NAMED WEB ENTITY EXTRACTORS

The search engines return the result web pages to the users according the keywords issued by them in their query. The keywords present in the query are matched with the indexed documents present in the search engine repository and the matched documents are returned. But the search engines are not always able to capture the intent of the user. This becomes more difficult when the intent of the user is not mentioned in the user's query. This becomes even more complex when the user is looking for a named entity. The queries fired on the search queries can provide a lot of useful information. The named entities are the real-life objects such as person, company, book etc. The term "Named Entity", now widely used in Natural Language Processing, was coined for the Sixth Message Understanding Conference (MUC-6) (62).For example, in the sentence "Narender Modi is the prime minister of India", Narender Modi and India are the named entities as they refer to some specific real world object. But prime minister is not a named entity as it can refer to the many real life objects in the real world. It happens because the different users can be looking for the same named entity but by using the different queries. The intent of the user is not considered while processing the query and fetching the results. To accomplish the task for collecting and understanding web information about the named entities, most of the search engines need human intervention. As per the study carried out by Microsoft about 71% of queries contain name entities [63]. So, to enhance the user's experience of search in WWW, the search engines have incorporated a lot of supervised and unsupervised techniques to extract the web entities and the named entities. Both the structured and the unstructured data from the web pages is processed to extract the entities. This information can be organized to knowledgebase [64] which can impact and improve the performance of a wide variety of applications such as parsing, conference resolution, and entity linking. Typically, researchers focus on extracting entities for movie, music, person, location, and organization names from natural language texts [65]. A lot of work has been done

in the recent years to fetch the named entities from the web. The techniques and algorithms developed to extract the named entities from the result web pages are discussed below.

Zaiqing [66] has extracted structured entities, named entities, entity facts and relations from the web. A new framework "iknoweb" has been proposed which is an interactive knowledge mining framework for entity information integration as shown in Fig 2.23.

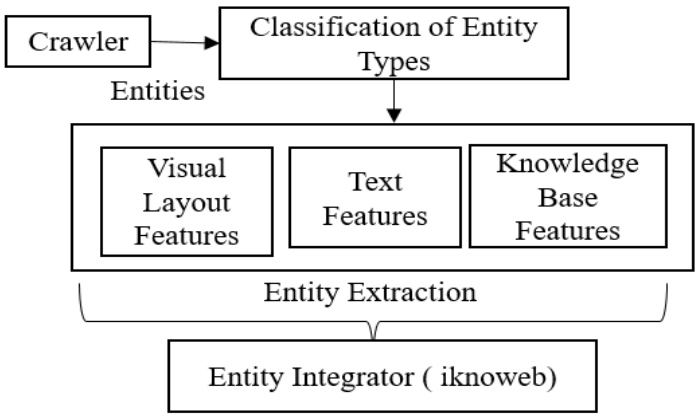

Figure 2. 23. The working of Iknoweb.

The entity search engine has been developed which crawls and extracts the entities from the web. The entity extraction search engine performed the following steps.

- Crawled web pages related to the targeted entities.
- The crawled data has been classified into different entity types such as papers, authors, products etc.
- Entity Extraction has been done from the web data using the visual based web entity extraction. The features for extracting the entities has been categorized in to three types.
    - Visual Layout Features

      Web pages have many explicit or implicit visual separators such as lines, blank area, image, font size, colour, element size and position. These features have been used to extract the entities.
    - Text Feature

      The natural text segments of a web page has been used as atomic labelling units. The long text sentences /paragraphs within the webpages have been further segmented into fragments using the semi-CRF algorithm.

- Knowledge base Features

  Some structured information of web entities has been stored in knowledge base. This information will be used further to improve the accuracy in the entity extraction procedure. When extracting the entities, if the details about them was found in the knowledge base then it helped in identifying the elements easily. So, information about the entities has been extracted accurately by this knowledge base feature.

- Entity Integration

  The details of one entity are found on multiple web pages. All these knowledge pieces extracted from the different web pages has been integrated. For that a novel framework "iknoweb" was introduced. It used a crowdsourcing approach which combined both the power of knowledge mining algorithms and user contributions.

Peter [67] extracted the entities from the unstructured information from the Wikipedia. The RDF triples were created from the unstructured Wikipedia text. These triples described the properties of entities and the relations between the entities. A new framework has been introduced to carry out an end-to-end extraction of DBpedia triples from the unstructured text. The input to the system was the Wikipedia articles and the output was the entities in the form of DBpedia triples. The system has been divided into two phases.

1) Wikipedia Article Processing

The Wikipedia articles have been processed by using the following steps

- WikiMarkup Filtering

  The main objective of this step was to extract the plain text for further processing. Wikipedia articles contained the special mark-ups called wiki text or wiki mark-up. These helped in hyperlinking with the other Wikipedia articles. So, this component eliminated all the comments, blocks that contains links and references.

- Wikipedia Link Extractor

  It extracted all the Wikipedia links from the article and stored these links along with their respective position of mentions in the article.

- Semantic Parsing

  The meaning of a sentence has been represented by a set of predicates and arguments. They have used the Athena framework. This framework has been

used for parallel semantic parsing of unstructured text. The system has used a high-performance multilingual semantic role labeller.

- Conference Resolution

  This module has used the conference solver to link the mention of entities in the different parts of the text. It discovered and then linked anaphoric phrases to their antecedents to create chain of referring mention. It helped in grouping entity action and properties described in different parts of the article. They also stored the named entity classification created by pipeline. Named entity classes have been used to filter named entity links having high conflicting ontology classification.

- Named Entity Linking

  To extract the entities the most important step is grounding of named entities to unique identifiers. They have used wikifier to link unannotated named entity in the Wikipedia articles to the corresponding DBpedia URI.

- Ontology Mapping

  In the semantic parsing done in step 3 the sentences were annotated with predicate-argument structure called the role sets. The objective of ontology mapping was to map the predicate and arguments roles from the Prop Bank to the DBpedia properties.

2) Entity Extraction

The predicate-arguments created during the semantic parsing were searched to find the named entity links corresponding to RDF subjects and objects. Entity extraction used the following to extract the entities.

- Mentions discovered in the conference solver.
- Links extracted by the Wikipedia link extractor.
- Named entity in the Wikipedia articles linked to the corresponding DBpedia URI using the wikifier.

It extracted the RDF subjects and RDF objects. To find the RDF subjects PropBank has been used. PropBank used A0 as the argument describing agents and A1 described the entities undergoing a state of change or being affected by an action. Arguments labelled as A0 and A1 were considered containing the RDF subjects and were consequently searched for named entity links. After extracting the RDF subject extraction object extraction was done.

Nathanael [68] have presented a novel approach to identify web objects that find relationships between the user's action and linguistic information associated with web objects. A new approach "WebLearn" has been proposed, a technique to identify the web objects based on unifying the semantic content of a user's utterance with the semantic content of the text written surrounding web objects as shown in Fig 2.24.

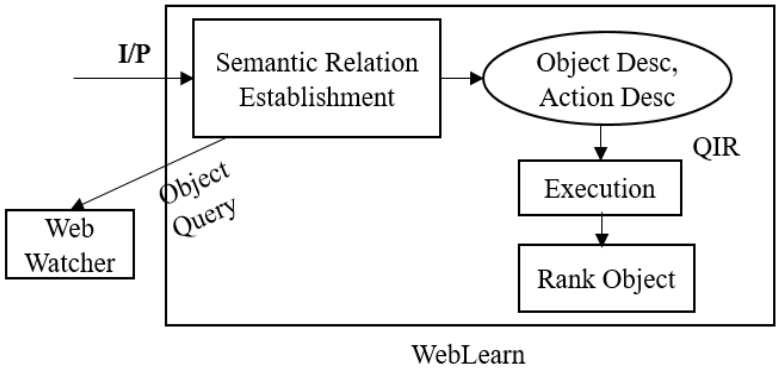

Figure 2. 24. Working of WebLearn.

The web objects referred to text fields, links (actions), buttons etc. The web objects contained the linguistic information that can be unified with the user's reason for selecting the objects. This helped in identifying the meaning of the object. The user goals were considered important. It learned how web objects relate to user's tasks. The main task of WebLearn was to associate the attributes of the web objects with the semantic representations of the user's utterances. The input to WebLearn was a 2- tuple: an action description and a web object ID. The action description was a feature-value structure based on knowledge representation. WebLearn interacted with the Webwatcher to retrieve information about the object. WebLearn then searched the object description to find the semantic association between the attributes and the current action. WebLearn parametrized both the object description and the action description where relation was found, creating an Object ID Rule (OIR) to associate the action with object. Action description was used as context for identifying the objects in the future. Object Identification Rule has two parts.

- Action description
- Object Description

Object description contained list of parameter variables and string relation. The string relations described the relationship between the parameter in action description and linguistic realization of semantic concept in the object description. String relation

allowed WebLearn to use linguistic information about the object that is not always visible to the user, but may be present due to human creation of the web pages. WebLearn was able to execute any rule that unify with a semantic action description input. The input action description was unified with all the WebLearn's OIR actions. All the features were not required, while doing the unification. But a penalty was added for the missing feature. When the variables unified, the actions were considered match and the extra features found were penalized in a similarity matrix. This matrix allowed for flexible action inputs and if multiple actions match, the one with the least edit distance was chosen. The variables binding from the action unification were then bound to the object description in the matched rule. Hence a new object was created which was used to query Web watcher for all the matching objects.

Ke Zhai [69] proposed an unsupervised approach based on the adaptor grammar to automatically identify the brand and product entities from a large collection of web queries in online shopping domain. The three different sets of grammar rules to infer query structures and extract brand and product entities have been proposed as shown in Fig 2.25. Their focus was to automatically extract the brand and products from web-scale online shopping queries. The proposed work was unsupervised so it does not require the human intervention and external resources. The noises were also handled. It was data driven so the details of the products and the brands were not known a priori. The entity extraction was composed of the following three steps.

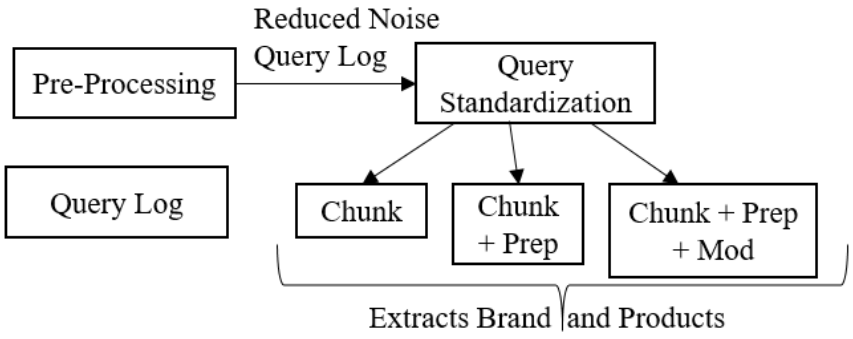

Figure 2. 25. Entity Extraction using Adaptor Grammars.

1) Pre-processing

The pre-processing has been done to remove the noises such as the misspelling or the noises because of the tokenization. To reduce such noises, they have collected non-conventional stop words frequently used in online shopping queries such as sale, deal, promo, price-range, cheap, discount, working condition, quality control etc.

2) Query Standardization

The noise is introduced when there are set of queries containing the same bag of words i.e. one unique query family. To reduce this noise query standardization has been done. The grouping of all the word orderings in a query family to the most frequent one was done. The frequency of every word ordering has been counted and collect distribution (w_i: c_i) has been done, where c_i was the frequency associated with word ordering w_i. The word ordering with the highest frequency count was found w=arg max w_i c_i and grouped the query family to word ordering ˆ w with frequency Σi c_i. This improved the parsing performance.

3) Grammar Rules

Three set of grammar rules have been proposed to infer query structures and extract brand and product entities. The first set of grammar rules decompose the query into product, brand or a combination of both. The underlined nonterminal refer to the adaptors. This grammar has been called as the "chunk" grammar as shown in Fig 2.26.

Query → Brand Product → Words
Query → Product Words → Word Words
Query → Brand Product Words → Word
Brand → Words Word → ...

Figure 2. 26. The Chunk Grammar Rules.

The second set of grammar rules decomposed query to brand, product, both or a combination of prepositional phrase. This grammar has been called as "chunk + prep" grammar as shown in Fig 2.27.

Query → Brand Brand → Words
Query → Product Product → Words
Query → Brand Product Words → Word Words
Query → Query PrepPhrase
Words → Word
PrepPhrase → Prep Words Word → ...
Prep → for|to|by|with|at|.

Figure 2. 27. The Chunk+Prep Grammar Rules.

The third set of grammar rules decompose query to brand, product, a combination of prepositional phrase, or a combination of modifier information. This grammar has been named as "chunk+prep+mod" grammar as shown in Fig 2.28.

Query → Brand Brand → Words
Query → ModProd Product → Words
Query → Brand ModProd Modifier → Words
Query → Query PrepPhrase
Words → Word Words
ModProd → Modifier Product
Words → Word
PrepPhrase → Prep Words Word → ...
Prep → for|to|by|with|at|...

Figure 2. 28. The Chunk +Prep+Mod Grammar Rules.

NERQ-2S [70] is a two-step process. The two steps are given below.

Step 1.

The named entity terms were differentiated from other types of tokens such as words, numbers etc. based on Conditional Random Annotated (CRF) data as shown in Fig 2.29.

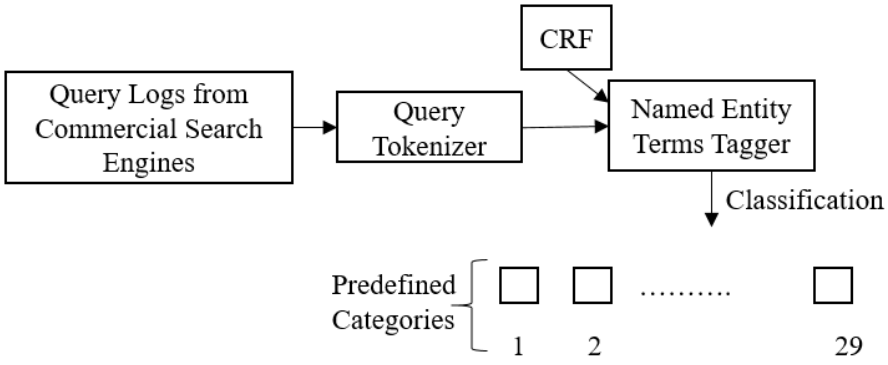

Figure 2. 29. Architecture of NERQ-2S.

Step2. NERQ-2S incorporated the output of the above CRF in to a new CRF as a feature. This 2nd CRF assigned each term within the query to 1 out of 29 predefined categories.

Xiaoxin [71] proposed a new approach for building a hierarchical taxonomy of the generic search intents for a class of named entities ex. Music, movies etc. The proposed

architecture found the phrases representing generic intents from the user queries and organized these phrases into a tree. The phrases which showed equivalent or similar meaning were on the same node and the parent-child relationships of the nodes represented the relationships between search intents and their sub-intents. Three methods have been proposed for building the tree. All the methods were based on the query logs. The main objective was to organize the generic search intents for a class of entities into a taxonomy according to the relationships between different intent. The working of the proposed architecture was divided into three parts as shown in Fig 2.30.

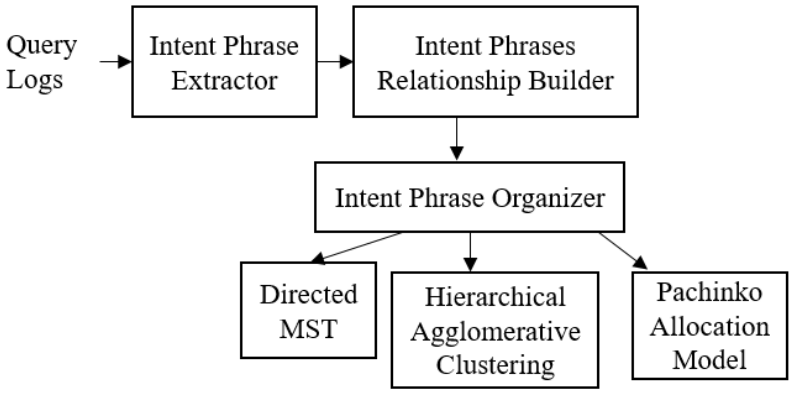

Figure 2. 30. Steps of Building Taxonomy of Search Intents.

1) Find the generic search intents

All the intent phrases that represent generic search intents and involves the entities of a certain class were selected. Many users express their intents in their queries explicitly, so most generic intent can be captured using all the intent phrases. If the intent was not expressed explicitly in the query, then to make sure an intent phrase represents a generic intent for different entities, the intent phrase should appear with at least θ entities in queries. θ was taken as 5.

2) Identify the relationships between the different intents

The relationships between the intents of the different queries was established first. Let q1 and q2 be the two queries. The intents of q1 and q2 was same if the information that can satisfy the needs of q1 can also satisfy the needs of q2. So q1 and q2 were likely to be equivalent of the intents of each of them belongs to those of the other. If the relationship between the intents of the different queries was estimated by their clicked URL's, then the relevance of a URL for a query was found, which was based on the query logs. The relevance of a URL 'u' for a query 'q' was defined by the equation 2. In equation 2 rel(q,u) was the estimated probability of 'u' being clicked when it was

viewed for ‘q’, ε was the soothing factor, click(q,u) was the number of times a user clicked ‘u’ for the query ‘q’ and skip(q,u) was the number of times user skipped ‘u’ for the query ‘q’.

$$rel(q,u) = \frac{click(q,u)}{click(q,u) + skip(q,u) + \varepsilon}.$$

(2)

For the two queries q1 and q2 the degree of search intents of q1 was being included in those of q2 was calculated by the equation 3.

$$d(q_1 \subseteq q_2) = \frac{\sum_{u \in U(q_1)} click(q_1,u) \cdot rel(q_2,u)}{\sum_{u \in U(q_1)} click(q_1,u)}.$$

(3)

d(q1⊆q2) was the average relevance to q2 of each clicked URL of q1, and d(q1⊆q2) ∈ [0,1]. For a class of entities E, and two intent phrases w1 and w2, the degree that the search intents of w1 were included in those of w2 was defined by the equation 4. d (w1 ⊆ w2) was the weighted average degree of the intents of a query containing w1 being included in those of a query containing w2, if these two queries contain the same entity. d (w1 ⊆ w2) ∈ [0,1] for any w1 and w2. If both d (w1 ⊆ w2) and d (w2⊆ w1) were high, then the intents of w1 and w2 should be very similar, and w1 and w2 were likely to be equivalent. If d (w1 ⊆ w2) was high but d (w2 ⊆ w1) was low, then w1 should correspond to a sub-concept of w2.

$$d(w_1 \subseteq w_2) = \frac{\sum_{e \in E, f(e+w_1)>0} d(e+w_1 \subseteq e+w_2) \cdot f(e+w_1)}{\sum_{e \in E, f(e+w_1)>0} f(e+w_1)}.$$

(4)

3) Organise the search intents relationships into a hierarchical structure

The intent phrases extracted above were organized into a hierarchical form i.e. tree. The root node of the intent tree did not contain any intent phrase. The nodes in the intent tree contained the intent phrases with the same or similar search intents. The child nodes of any node ‘n’ contained the intent phrase that represents the sub-concept of the intent phrase of node ‘n’. Three approaches were used for organizing the intent phrases in to a hierarchical form. The three approaches are given below.

- Method based on directed maximum spanning tree.
- Method based on hierarchical agglomerative clustering
- Baseline method based on pachinko allocation models

2.9. CONCLUSION

This chapter discussed the evolution of internet and the techniques to extract the hidden data from the internet. The study reflects that the techniques to uncover the hidden data can be categorized as named entity extraction, query interface extraction, query interface integration, hidden web data aggregation. The combination of these techniques should be incorporated to analyse and integrate the hidden web. In the proposed work DQPHDE, the academic search engine has been developed. The end users are interested in both the surface web and the hidden web for the information extraction of their desired domain. None of the work discussed above processes and integrates the data from both these portions of WWW i.e. hidden web and surface web. Most of the recent techniques do not make use of artificial intelligence techniques to extract the hidden web data. If the artificial intelligence is incorporated in the techniques of hidden web data extraction then the task of hidden web data extraction can be done easily as done by the users. In the proposed work all the limitations are addressed and a new solution is proposed to help the end users. The next chapter throws some light on the techniques and algorithms incorporated in the proposed technique DQPHDE.

CHAPTER 3

RELATED WORK

3.1. INTRODUCTION

The data stored in the hidden web database is of very high quality and quantity. The traditional search engines are not able to index this data efficiently. So when the users issue their query on search engines, the results shown to them are of low quality and only a small section from WWW is shown to the users as output. In order to improve the user's searching experience the data from all section of WWW should be extracted and should be displayed to the users. The data on WWW is very humongous. So a lot of processing and extraction algorithms need to be applied on this data for the better output. In the proposed work DQPHDE a lot of information extraction algorithms, clustering algorithms and block section extraction algorithms have been used. This chapter throws some light on the different techniques and algorithms used in the proposed work DQPHDE. The different techniques and algorithms used in DQPHDE are semantic text mining, clustering, fuzzy rule based system, winnowing algorithm [102] and vision based page segmentation algorithm (VIPS) [113]. The semantic text mining, a kind of information extraction technique is used in DQPHDE, for the processing and extraction of information from the unstructured web pages which is discussed in section 3.2. A novel graph based clustering algorithm is used for the clustering of the research articles which is being discussed in section 3.3. The Winnowing Algorithm is used in the proposed work for finding the similarity of different key features which is discussed in section 3.4. VIPS algorithm is used for the extraction of different sections from the web pages is being discussed in section 3.5. The fuzzy rule based system is used in DQPHDE for the automatic discovery of the web forms from the web pages is being discussed in section 3.6.

3.2. TEXT MINING

Text mining is the process of extracting knowledge from unstructured or semi-structured text. Text mining is similar to data mining in the manner that both of them helps in the extraction of knowledge. However in the case of data mining the knowledge extraction is done from the structured databases. While in the case of text mining the knowledge extraction is done from the unstructured or semi-structured documents. As there is so much advancement in technology there is large amount of information in the

digital form and most of the information (around 85%) is in unstructured text form, so a lot of research has been done in area of text mining for information extraction [78-83]. It is becoming very important to build new algorithms and propose new approaches to fetch maximum useful information out of that unstructured text data. Text Mining is the best way to fetch potentially useful information from large set of text documents. The text mining systems take the raw text documents as an input and generates patterns, trends or knowledge as an output. There are four modules in a text mining system namely pre-processing, core mining operations, presentation layer and browsing functionality, and refinement techniques as shown in Fig 3.1.

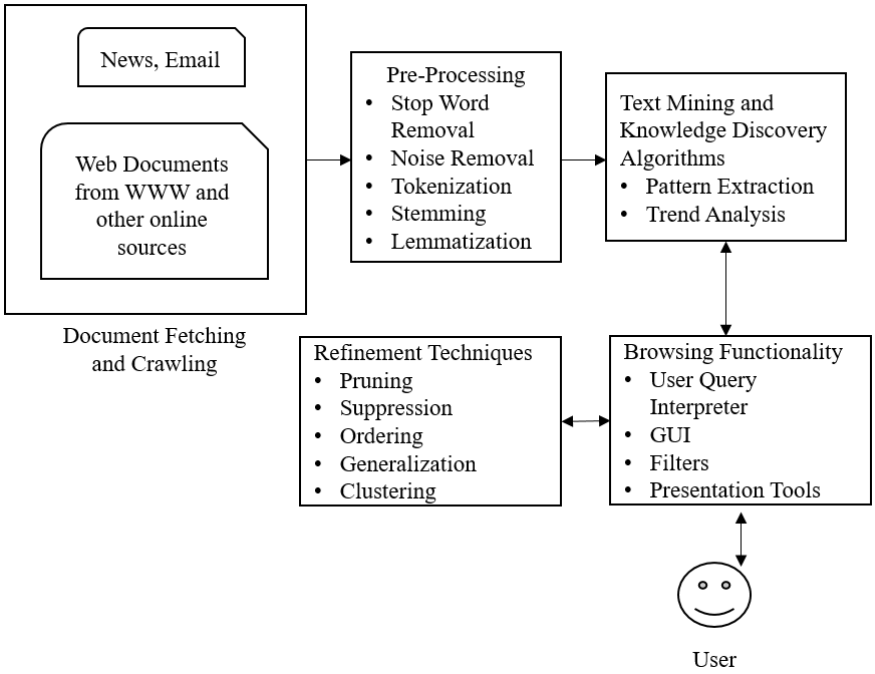

Figure 3. 1. The Framework of a General Text Mining System.

- Pre-Processing

Pre-Processing of text plays a very vital role in text mining as it prepares the text documents for further processing. It removes noises and converts the text documents into canonical format. The tasks carried out in pre-processing step are tokenization and normalization. In tokenization the text from the text document is taken as input and a stream of words is produced as the output. Tokenization is also called as lexical analysis. Tokenization first of all divide the entire text into sentences. Then the sentences are further broken into words. Normalization converts all the words extracted by tokenization into the same form. All the words are converted into either lowercase or uppercase. The stop words, punctuation symbols, white spaces are removed. These

words are removed from text documents as they are not measured as keywords in text mining [84]. The numbers found in tokens are converted to text. Apart from these actions, normalization performs stemming and lemmatization. In stemming the various forms of a single word are converted into one form. For e.g. run, ran, running all these three words are converted to run. In lemmatization the words are analysed morphologically. To do the lemmatization the detail dictionary of that language should be present. When users issue their queries they are not expecting the exact match of the keywords. They want the words relevant to the query keywords to be included in the search. So lemmatization is done.

- Core Mining Operations

The core mining operations includes the algorithms which help in the discovery of patterns, trends and knowledge extraction from the text documents. The types of pattern or trends found in text documents are distributions or proportions, frequent or near frequent sets, and association.

- Presentation Layer and Browsing Functionality

In text mining system it refers to interface between the knowledge discovery algorithms and the user. By the help of this interface end users issue their queries to system, view the results and extracts the knowledge or pattern.

- Refinement Techniques

The refinement techniques helps in the removal of redundant information. It also has suppression, pruning, ordering and clustering algorithms. These techniques are also called as post-processing.

Text Mining is not that much effective and cannot achieve high accuracy as compare to “Semantic Text Mining”. The data on WWW is unstructured and the machines are not able to understand this data. In order to make the text of WWW machine readable it has to be converted into a format which is understandable by the machines. Ontology provides the methodological provision and support to represent and organize semantic information. In the proposed work DQPHDE the Semantic Text Mining is being used. The conventional text mining does not help in establishing the relationship between the words. So in the proposed work the text from the web document is converted into OWL (Web Ontology Language) descriptions. By the help of Web Ontology Language the classes found in web documents are created and the relationship between these classes is established. The classes created using the OWL are machine readable and the

documents can be processed by the machine. In the semantic text mining the words are extracted along with their meaning and if the relationship is there between the words then that can be presented easily here.

3.2.1. Semantic Text Mining

Semantic text mining goals at merging two advanced technologies i.e. semantic text and text mining. Recently, text mining researchers are more attracted towards semantics to improve the results of text mining. In pre-processing step of text mining data preparation can use some sort of semantic or ontology of text collection. And in pattern extraction semantic data can be used to prepare the model generation and to polish it and in last step i.e. post-processing, results can be evaluated based on semantics aspects. Proper use of semantics with text mining can result out as better results for many of the applications.

In the proposed work DQPHDE the new technique of text mining using semantics has been proposed. The web documents of a particular domain are extracted from WWW. The text from web documents is taken as input and a semantic graph is created using the semantic based text mining. The semantic graphs of different web documents are compared and the summarization and integration of text from multiple web documents is done in the proposed system.

3.3. CLUSTERING

Clustering is defined as the process of making group of identical items. These groups are called as clusters. The items in different clusters are different from each other and based on some features items in the same cluster are identical to each other. Different clusters may be formed using same data set by applying different clustering methods [85]. Clustering is an important unsupervised form of classification technique of data mining [85, 87]. Clustering is a very important technique in data mining applications such as information retrieval, web analysis, CRM (Customer Relationship Management), medical etc. Data clustering is recognized as an important area of data mining [88]. The main uses of clustering are given below.

1. With the growth of internet, data storing techniques, images, videos etc., huge size and high dimensional data sets have been created. It is not easy to understand this bulk of data. So clustering helps in understanding the data with ease. If there are some glitches in the data then these glitches can be easily

detected by the help of clustering techniques. When the data is so huge then it is difficult for the users to identify and extract the important features. Clustering helps in the identification of those features which cannot be observed by users.

2. Clustering also helps in the natural classification. Clustering can help in grouping organisms by finding how similar the organisms are. This type of relationship finding is also known as phylogenetic relationship.
3. Clustering can also be used for compressing, organizing and summarizing the data.

Difference between Clustering and Classification

Both clustering and classification are the types of learning methods which are used to group the data or items having similar properties. But these two methods are different in terms of data mining. The differences between them are given below in Table 3.1.

Table 3. 1. Difference between Classification and Clustering

Classification	Clustering
Classification is supervised machine learning technique.	Clustering is unsupervised machine learning technique.
In classification the new object is always assigned to one of the pre-defined class or category. So categories in classification are pre-defined and fixed.	In clustering the classes or categories are not pre-defined. This is the reason that clustering is known as unsupervised learning technique.
Training is provided in classification.	Training is not available in clustering.

Classification of Clustering Techniques

According to Chen [89], unsupervised clustering is divided into hard clustering and soft clustering.

1. Soft Clustering: Soft Clustering is also known as the Fuzzy clustering or Overlapping clustering or Non-exclusive clustering. In soft clustering the

unclear cluster sets are used. Here the data/object/item can be placed in more than one cluster.

2. Hard Clustering: Hard Clustering is also known as the Exclusive or Non-fuzzy clustering. In hard clustering the cluster sets are clear. The data/object/item is assigned to exactly one cluster.

Types of Clustering Algorithms

Most clustering algorithms are based on two popular techniques known as hierarchical and partitioning clustering [94].

- Partitioning Algorithm: In partition clustering the data analyst specify the number of groups or clusters to be created in advance. The clusters created in portioning algorithms are non-overlapping or disjoint. Partition clustering algorithms are generally iterative algorithms that converge to local optima [90]. For e.g. there are "N" objects and the number of pre-specified partitions a "k". Then the partitioning algorithms will create "k" clusters such that each cluster has minimum one object and each object belongs to one cluster only. Iterative K-means algorithm [91] is the most popular partitioning clustering algorithm. The other partitioning clustering algorithm is Fuzzy C-means which is the fuzzy form of K-means. It was developed by Bezdek [92]. Another partition based clustering algorithm is Gaussian Expectation-Maximization (EM) algorithm [93].
- Hierarchical Algorithm

  Hierarchical clustering [95] was introduced by Johnson in 1976. In hierarchical clustering the similar objects are placed in same clusters. The tree structure, called as dendrogram, is used for representing the clusters. The leaf nodes in this dendrogram are the subset of the input data set. Hierarchical clustering is of two types.

  Agglomerative Clustering: It uses the bottom up methodology. In agglomerative clustering, initially all the objects from the input data set are placed in different clusters. Then the two clusters which are identical are combined or merged. This procedure is repeated until some end condition is met. The two most popular agglomerative hierarchical algorithms are the single link [96] and complete link [97] algorithms.

Divisive Clustering: It uses the top-down methodology. In divisive clustering, initially all the objects from the input data set are placed in one single cluster. Then the cluster is split in to two. One of the cluster is then taken and is split into smaller clusters repeatedly until some end condition is met. Bisecting K-means is a type of divisive clustering algorithm.

In the proposed work DQPHDE, the graph and relationship based clustering technique has been proposed. The proposed graph and relationship based clustering technique is based on hierarchical divisive clustering. So all the nodes of the input graph are placed in one cluster in the proposed technique. The top-down methodology is being used in the proposed technique. It will cluster the web documents. Document clustering assists in retrieval by creating links between related documents, which in turn allows related documents to be retrieved once one of the documents has been deemed relevant to a query [98]. In DQPHDE the clustering technique uses the relationship strength between the web documents as the distance matrix. By the help of this distance matrix the web documents will be clustered.

3.4. DATA SIMILARITY ALGORITHMS

In the world where most of the information is digital, documents are easy to copy. The digital signatures are put on these documents for the security reasons. But these days removing the embedded fingerprints or any signature is easy, which makes the detection of piracy quite difficult. Anyone can just copy or retype the document to make their own. Hence the data similarity algorithms were developed to avoid the piracy of digital data. The data similarity algorithms are used in many pattern recognition applications that involve clustering, classification, recognition, or retrieval. Similarity measures vary depending on the data types used [99]. The data similarity algorithms are of three types namely string-based, corpus-based, and knowledge-based algorithms.

- String-based Algorithms: The string based algorithms finds the similarity between the web documents by using the strings or the characters found in the document. The string based methods can be further divided into two.
    - Character-based Algorithms: These algorithms finds the difference between two strings by applying some operations on the characters found in the string. The examples of few character-based algorithms are shown in Fig 3.2.

- Term-based Algorithms: In the case of the term-based algorithms the frequency of the terms found in the different documents is used to calculate the similarity between the documents. The examples of few character-based algorithms are shown in Fig 3.2.

- Corpus-based Algorithms: In corpus-based algorithms the similarity between the two words is found by the help of a large database containing words. The examples of few character-based algorithms are shown in Fig 3.2.
- Knowledge-based Algorithms: In knowledge-based algorithms the similarity between the web documents is found by the help of the information derived from semantic networks. The examples of few character-based algorithms are shown in Fig 3.2.

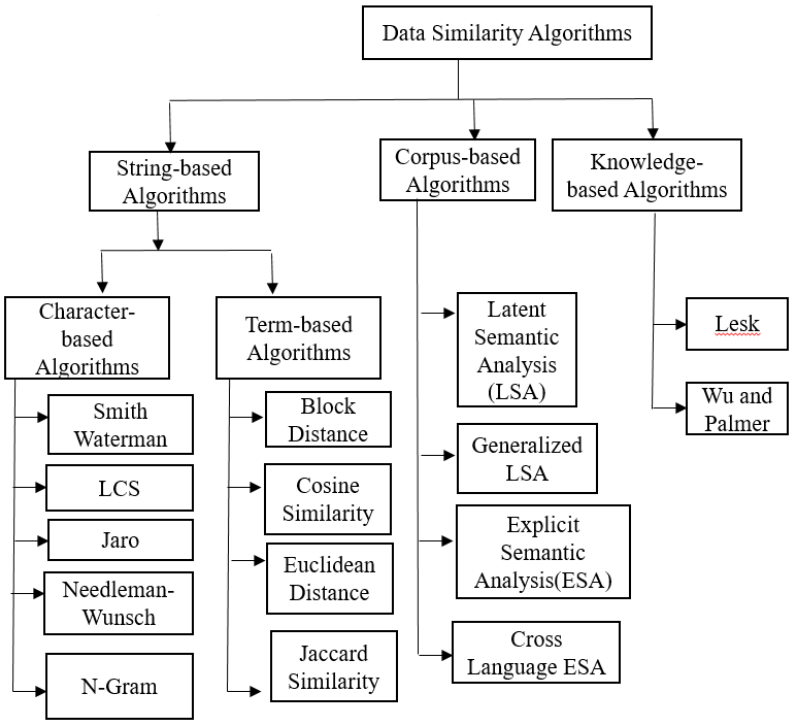

Figure 3. 2. The Different Types of Data Similarity Algorithms.

In the proposed work "DQPHDE" the web documents containing the research articles are clustered. For clustering the research articles, the similarity of key features found in research articles has to be found. For that data comparison fingerprint algorithm and winnowing algorithm have been used. Fingerprint and the Winnowing algorithms are the character based algorithms which make use of N-gram. So the details of these algorithms have been shown below in the next section.

3.4.1. Fingerprint Algorithm

Fingerprint algorithm [102] is a type of character-based algorithm which makes use of the N-grams. Document fingerprint algorithm helps in the detection of full and partial copies of the documents. In the fingerprint whole documents are not stored for the comparisons instead small sketches of the documents are stored. By comparing the sketches of two documents, the substantial overlap [100] between them is found.

According to Dittmann [101], a digital document fingerprinting scheme consists of a number of marking positions in the document. Fingerprint search in a document can be done by using a hash function in N-Gram of a document. N-Gram is a substring of length k by combining all the tokens and symbols in the document. The steps used in the fingerprint algorithm are given below.

1. Remove all the spaces from the input string
2. The n-gram sequence of characters are created from the string derived from step 1.
3. MD5 (Message Digest 5) is applied to the n-gram sequences of characters.
4. Fingerprint is calculated using MD5 mod p=0
5. Fingerprints are matched using the Dice Coefficient
6. Similarity value is calculated.

The example of fingerprint algorithm is shown in Fig 3.3 [102].

(a) Some Text.
A do run run run, a do run run
(b) The text with irrelevant features removed.
adorunrunrunadorunrun
(c) The sequence of 5-grams derived from the text.
adoru dorun orunr runru unrun nrunr runru unrun
nruna runad unado nador adoru dorun orunr runru
unrun
(d) A hypothetical sequence of hashes of the 5-grams.
77 72 42 17 98 50 17 98 8 88 67 39 77 72 42 17 98
(e) The sequence of hashes selected using 0 mod 4.
72 8 88 72

Figure 3. 3. Example for Fingerprint Algorithm.

The fingerprint algorithm has a disadvantage, that certain times no fingerprint is found. It happens when there is not any hash values which satisfies the 0 mod 4. The solution to this problem is provided by the winnowing algorithm [102]. The winnowing algorithm is an extension of the fingerprint algorithm.

3.4.2. Winnowing Algorithm

Winnowing algorithm [102] is also a character-based algorithm which uses the N-gram. The drawback of fingerprint algorithm is resolved here. To remove the problem found in fingerprint algorithm, winnowing algorithm makes the use of window concept. The starting three steps of the winnowing algorithm are same as that of the fingerprint algorithm. In the next step the window of hash functions is created as shown below.

1. Remove all the spaces from the input string
2. The n-gram sequence of characters are created from the string derived from step 1.
3. MD5 (Message Digest 5) is applied to the n-gram sequences of characters.
4. From the hash values calculated above, the window of hash values of length 4 is created.
5. The fingerprints are calculated using the winnowing algorithm. The winnowing algorithm selects the hash values from each window using the two rules given below. This hash value set then becomes the fingerprint.
   5.1. From each window minimum value is chosen.
   5.2. If there are more than two hash values then the smallest hash value from the right side of the window will be selected.
   5.3. If hash value is selected which has a position overlap the position of hash value selected before, that value is chosen once.
6. Fingerprints are matched using the dice-coefficient.
7. Based on the dice-coefficient value the similarity is calculated.

The example of winnowing algorithm is shown in Fig 3.4 [102].

(a) Some Text.

A do run run run, a do run run

(b) The text with irrelevant features removed.

adorunrunrunadorunrun

(c) The sequence of 5-grams derived from the text.

adoru dorun orunr runru unrun nrunr runru unrun

nruna runad unado nador adoru dorun orunr runru

unrun

(d) A hypothetical sequence of hashes of the 5-grams.

77 72 42 17 98 50 17 98 8 88 67 39 77 72 42 17 98

(e) Windows of hashes of length 4. (77, 74, 42, **17**) (74, 42, 17, 98) (42, 17, 98, 50) (17, 98, 50, **17**) (98, 50, 17, 98) (50, 17, 98, **8**) (17, 98, 8, 88) (98, 8, 88, 67) (8, 88, 67, 39) (88, 67, **39**, 77) (67, 39, 77, 74) (39, 77, 74, 42) (77, 74, 42, **17**) (74, 42, 17, 98)

(f) Fingerprints selected by winnowing. 17 17 8 39 17

(g) Fingerprints paired with 0-base positional information. [17, 3] [17, 6] [8, 8] [39, 11] [17, 15]

Figure 3. 4. Example of Winnowing Algorithm.

3.5. WEB PAGE SEGMENTATION

With the advancement of the internet, new techniques are developed to create web pages. These days the web pages are created by the help of content management systems (CMS). Joomla, WordPress, Drupal, and SilverStripe etc. are few examples of CMS. Because of the development of CMS, the automatic extraction of different segments from web pages is not an easy task. The web pages have different sections which are not related to the actual content of a page such as navigation bar, contact details, advertisements, and comments by users etc. So in order to extracts the blocks or different parts from the web pages, the web page segmentation techniques were developed. Web Page Segmentation (WPS) is defined as the technique where the web page is divided into the blocks by the help of visual clues and semantic clues. WPS has many application areas such as mobile application [103], information retrieval [104], web achieving [105] etc. A lot of research has been conducted to automatically extract the segments from the web pages. A lot of problems are faced while extracting the blocks from the web pages [106]. These problems are discussed below.

1. The source code of web pages are intertwined. It leads to a fuzzy boundary between the source code and the visual layout of a web page.

2. The different web site developers have different approaches to create the web pages. So it leads to heterogeneous layout of different web sites.
3. In the process of analysing the structure of web pages, performance is a big challenge.
4. There a many dynamically generated web pages. So extracting blocks from such web pages becomes difficult.

The different web page segmentation techniques have been developed. Many researchers have used the tag information for the process of dividing the web page into blocks or segments. While many others have harnessed the visual structure of the web pages for extracting the blocks or segments from the web pages. Diao [107] have developed a learning based web query processing system. In the system the web page have been divided into segments by the help of few important tags such as <p> paragraph, <ul> unordered list, <table> table tag, headings <h1> to <h6>. According to Lin [108] the content of web page is mostly rendered in the <table> tag and its child tags. From the multiple table tags, the required table tag have been extracted by the entropy based technique. The <p>, <table> and <ul> tags have been used by Kaasinen[109] and Buyukkokten[110]. Embley [111] and Buttler [112] have used some heuristic rules to discover record boundaries within a page, which assist data extraction from the web page. Another algorithm developed for extraction of blocks from web pages was VIPS (Vision Based Page Segmentation). VIPS was proposed by Deng in 2003 [113]. VIPS have used the spatial and the visual clues for dividing the web page and extracting the underlying structure of web page. VIPS have used the top-down approach for segmenting the web page.

In the proposed work “DQPHDE”, to extract the different blocks the VIPS algorithm have been used. The blocks are extracted in DQPHDE for the extraction of the result section from the result web pages. The detail working of VIPS algorithm is discussed below.

3.5.1. Vision Based Page Segmentation Algorithm

The VIPS algorithm have used the visual clues and the semantic structure of web pages for the task of extracting the blocks from the web pages. The semantic structure is represented by a hierarchical or a tree like structure. The nodes in this hierarchical

structure represents the blocks. A web page in VIPS have been represented by a triplet shown in equation 1.

$$\Omega = (O, \phi, \delta) \tag{1}$$

Here Ω represents the web page, O represents the finite non-overlapping block set. Each non-overlapping block is a sub web page. O has been represented by the equation 2 shown below.

$$O= \{\Omega 1, \Omega 2 \ldots\ldots \Omega M\} \tag{2}$$

M is the total number of non-overlapping blocks in the web page Ω. ϕ is the set of tags or separators, which separates the different blocks for e.g. the vertical separator tags or the horizontal separator tags. The separator set is represented as shown below in equation 3. The separators are assigned weight based on their power of dividing the web page. Every separator in ϕ^n have same weight.

$$\phi= \{\phi 1, \phi^2 ..., \phi^n\} \tag{3}$$

δ represents the relationship between the different blocks in block set "O". This function δ returns the separator tags between the blocks. Let's assume there are two blocks $\Omega 1$ and $\Omega 2$ in a web page Ω. Then δ ($\Omega 1$, $\Omega 2$) will return some value if there is some separator tag between the blocks $\Omega 1$ and $\Omega 2$.

For example if we have a web page as shown in Fig 3.5, then the web page can be divided into blocks using the visual layout as shown in Fig 3.6.

Figure 3. 5. Home Page of Yahoo Shopping.

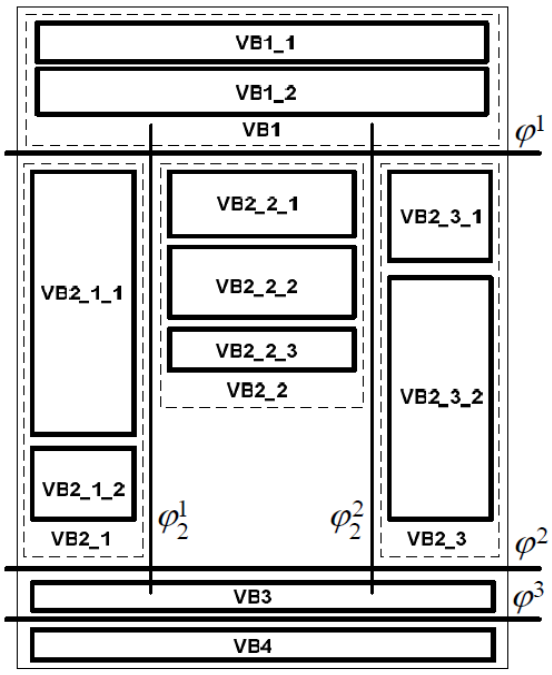

Figure 3. 6. Different Blocks of Web Page.

The segmentation is done in VIPS using three steps namely visual block extraction, visual separator detection and content structure construction. These steps are explained in detail below.

Visual Block Extraction: The web page is divided into blocks or sub web page. From these sub-web page, all the visual blocks will be extracted. From the web page DOM structure, the visual blocks are extracted. Every node in the DOM tree shows the visual block. If the node is big such as <p>, or <table> etc. then the node is further divided. As HTML is very flexible, so all the web pages don't follow W3C specifications, so DOM tree does not give the exact visual blocks always. So Degree of Coherence of each extracted block is set according to the intra visual distance. This process is repeated until all the visual nodes or blocks are found.

Visual Separator Detection: The visual blocks extracted by visual block detection are placed in the pool. From these blocks the visual separator is extracted. The visual separators are the horizontal or the vertical lines. After extracting the separators the weights are assigned to the blocks.

Content Structure Construction: After the extraction of separator and assigning the weight to the separators, the structure of the content is build. The separator which have the lowest weight are picked first. The blocks which are nearby these separators are then merged and new blocks are formed. The blocks are merged until the separators with the heighted weight are picked. The degree of coherence of new blocks created is set again based on the maximum weight of the separator in the block. It is then followed by the granularity check of each leaf node. If this granularity check failed then visual block extraction is done again and the content structure is reconstructed again for that node. If the granularity check is approved by all nodes then the final content structure of web page is obtained.

3.6. ARTICIAL INTELLIGENCE AND THE EXPERT SYSTEM

The word Artificial intelligence was first coined by John McCarthy in 1956 in the first conference on the subject in Dartmouth College. The basic idea of Artificial Intelligence is machines working or behaving like the humans. Alan Turing was able to prove through his tests that the "machines can think". These test were known as the Turing Tests. Artificial Intelligence is defined as a field that describes the skills of machine's learning as done by the humans and the skills of machines to respond

according to those learnings just like human beings. The aim of A.I. is to make computers or machines more intelligent and make them behave like humans. AI allows machines to process complex data and provide accurate information [114]. There are few differences between the traditional programming and the AI based programming. These differences are listed in Table 3.2.

Table 3. 2. Difference between Traditional Programming and Programming with AI.

Traditional Programming	**Programming with AI**
Traditional Programming is used to solve and give answers to the specific problems.	Programming with AI helps in solving the generic problems.
If the program is edited then it can lead to change in the structure of the program.	In AI the information is placed together which is independent of each other. So AI can easily absorb the changes done without affecting the structure of other piece of information
The changes or updates are not quick and easy. They may affect the program undesirably.	The updates are quick and easy here.

The use of Artificial Intelligence is increasing day by day. AI can be applied for the problem solving in wide variety of task domains as shown in work done by Elaine [115]. AI has grown so much that it is being used in different kinds of applications such as gaming, vision systems, speech recognition, handwriting recognition, intelligent robots etc. The tools developed using AI can be categorized as mundane tasks, formal tasks, expert tasks as shown in Fig 3.7 [116].

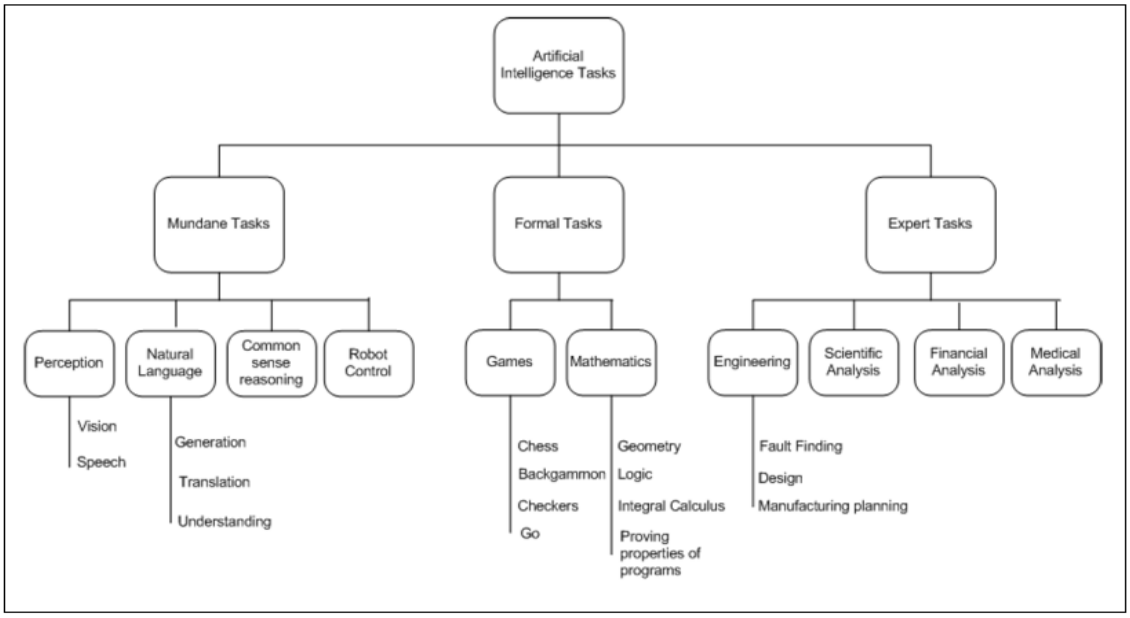

Figure 3. 7. Some task Domains of AI.

There are different areas of A.I. such as Expert Systems, Speech Understanding, Robotics, Natural Language Processing etc. From these areas the Expert System is growing rapidly and is being used in different applications. The Fuzzy Expert System is being used in the proposed work DQPHDE. Fuzzy logic is a data handling methodology that permits ambiguity. It captures and uses the concept of fuzziness in a computationally effective manner. Fuzzy Expert System uses the rules in the form of "if-else" structures. The expert systems can solve the problems of domains as done by the experts of those domains. Expert System is one of the noticeable area of AI. Expert System is defined as a program which uses the techniques of AI to match the behaviour and decision making capability of domain expert user to solve the problems of that domain. The expert system is able to make decisions like human by maintaining the database of knowledge. This knowledge is stored and coded in a form which can be used by the expert system for inference purpose. The source of knowledge added in the expert systems are the human experts, online sources etc. A lot of training is provided to the expert systems to acquire this knowledge.

3.6.1. Working of Expert System

The expert systems has two components the knowledge base and the consultation system. The consultation system comprises of inference engine, explanation facilities, trace facilities and user interface as shown in Fig 3.8. The knowledge base can be developed separately from the inference mechanism. There are several advantages of

developing the inference mechanism and knowledge base development separately. The first advantage is that the knowledge base can be corrected and updated without affecting the inference mechanism. Second advantage is that the knowledge base can be substituted with another domain's knowledge base and a new expert system can be created. The knowledge base consists of the domain-specific knowledge and the inference mechanism consists of the algorithms for deriving knowledge from this knowledge base.

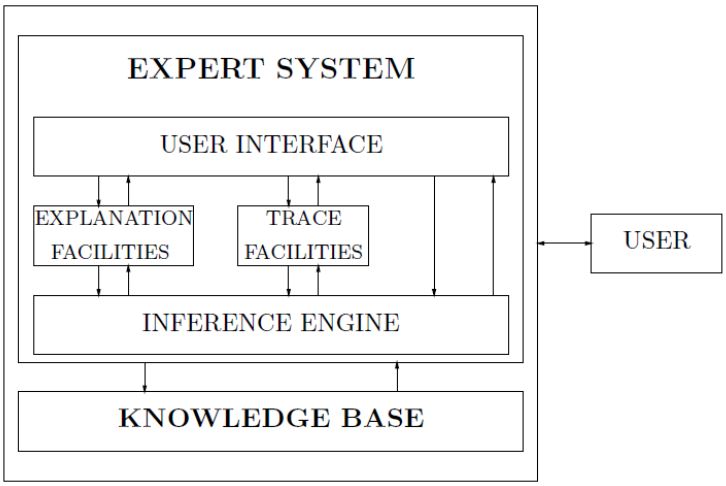

Figure 3. 8. Architecture of Expert System.

Knowledge Base

The knowledge base of a domain is created by using multiple knowledge sources such as human expert in the domain, internet, books and databases. The person creating the knowledge base is known as the knowledge engineer. The task of gathering and storing the knowledge for a particular domain is called as knowledge acquisition. Knowledge acquisition is the most important element in the development of expert system [117]. When the domain expert is interviewed to collect the information then it is known as knowledge elicitation. The knowledge is stored and coded in a form so that it can be easily understood by the computers. There are three knowledge representation methods used for storing the domain details. These three methods are given below.

- Logic

  Logic is one of the earliest method for the representation of knowledge and is used least for developing the expert systems. The drawbacks of logic are that inference in logic is a NP-complete problem and there are not so many successful techniques in logic for expressing the heuristic shallow knowledge.

The logics can be different types such as Prepositional Logic, First Order Predicate Logic, Modal Logic, Non-monotonic Logic etc.

- Production Rules

  Newell and H.A. Simon developed production system as a psychological method for representing human behaviour. Here human knowledge is presented as productions or rules. Many expert systems these days use production rules for the representation of knowledge. All the rules in the knowledge base of an expert system are called as rule base. For defining the objects or the domains the facts are there in the rules. Together these rules and facts make the knowledge base. In the proposed work DQPHDE, the production rules are used for defining the facts and rules. These facts and rules will help in extracting the query interfaces from the web pages.

- Semantic Nets and Frames

  In 1960's R. Quillian used the semantic nets or associative nets for representing English words with their meaning in terms of associative links to other words and creating a graph like network. Semantic Networks have been rarely used for creating the expert systems. The basic concept of a frame-based knowledge representation is that all the information related to classes and the relationship between the classes are stored in a frame. The set of frames holding the details of a particular domain are organized which is then called taxonomy. This taxonomy form helps in the automated reasoning known as the inheritance.

Consultation system

As discussed above the consultation system comprises of user interface, explanation facilities, trace facilities and the inference mechanism. User interface helps in the interaction of the user with the expert system in the form of question-answering sessions. The explanation facilities helps in asking questions such as how certain conclusions were drawn, why a particular question is asked or why other conclusions were not drawn during the consultation procedure. By using the trace facilities the reasoning behaviour can be followed one inference step at a time during the consultation. The inference engine consists of algorithms for updating the knowledge in the knowledge base.

3.6.2. Fuzzy Expert System

There is a lot of information available in WWW and the world. Most of the information available is fuzzy i.e. not certain and imprecise. But the human beings are able to process and handle this uncertain, fuzzy and vague information efficiently. The way human thinking and decision making happens in such cases can't be expressed clearly. In such cases can be expressed and measured using the statistical and probability theory. The fuzzy expert system helps to model such kind of uncertainty. Zadeh introduced the fuzzy systems in 1965 [118]. A fuzzy expert system is simply an expert system that uses a collection of fuzzy membership functions and rules, instead of Boolean logic, to reason about data [119]. The basic architecture of a fuzzy based expert system is shown in Fig 3.9. The fuzzy expert system has four components namely fuzzification interface, a fuzzy rule base or the knowledge base, inference engine and defuzzification interface.

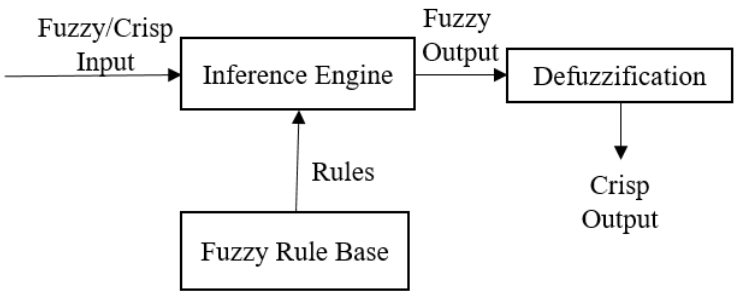

Figure 3. 9. Architecture of Fuzzy Expert System.

Knowledge Base

The knowledge base in the expert system has the rules and facts. The facts help in the domain description. The format of the rules is given below.

If a=low and B=high then c=medium

Here "a" and "b" are the input variables and "c" is the output variable. The fuzzy set is defined for each variable "a", "b", and "c". For ex: low, high and medium are defined for variable "a","b","c" respectively. The conditions are specified in "if" part of the rule. This part of the rule is known as the antecedent. When the conditions in the "if" are true then the rule is said to be triggered and the actions are fired. When the action is fired then the knowledge base can be updated sometimes. The "then" part of the rule is known as the consequent. The "then" part of the rule tells the actions to be fired when

the rule is triggered. These rules and the reasoning are the core of any expert system. The collection of these fuzzy rule forms the knowledge base of the fuzzy logic system.

Inference Engine

The inference engine finds the truth value for the antecedent of every rule and the consequent part of every rule if fired. The output is a fuzzy subset which is assigned to each output variable for each rule. Then by using the composition procedure the fuzzy subsets which were assigned to each output variable are combined together and a single fuzzy subset is created for each output variable. Then this fuzzy output is converted into a crisp output by the defuzzification. The final output of any fuzzy expert system is always a crisp output.

3.7. CONCLUSION

The chapter discusses the various techniques used in the proposed work. These techniques are used for the extraction of information from the web documents and finding the similarity among them. The data related to a topic is scattered among multiple unstructured web pages. In order to extract the summary and the unique text from these web documents semantic based text mining has been done. To cluster the research articles of a particular domain a new graph based clustering technique has been developed in the proposed work. To compare the different web documents, the different data similarity algorithms are there. The winnowing algorithm has been used to compare the web documents. The web pages are partitioned in to blocks by the help of the VIPS algorithm which uses the visual and the semantic clues. These blocks are further used for the extraction of web forms and the result sections from the web documents. For the detection of the web forms and the result sections from the web pages, the flavour of artificial intelligence has been incorporated in the proposed work. The fuzzy rule based expert system has been used to extract the web forms and result sections. In the next chapter, the proposed method “DQPHDE” has been explained in detail.

CHAPTER 4

DYNAMIC QUERY PROCESSING FOR HIDDEN WEB DATA EXTRACTION: THE PROPOSED WORK

4.1. INTRODUCTION

The rapid growth of internet is carrying complications with it in terms of creation of static web pages. To display enormous data on them website administrator has to create thousands of web pages to display data. However user is interested in viewing certain web pages of his/her interest. So, the web site administrator places the data in the database and this data is displayed on the web pages only on user's demand. The web site administrator creates query interfaces and users query fetches the data from the databases dynamically by filling required fields. To extract the desired information from different portions of web, the user issues the query on query interfaces of different web sites which consumes a lot of time and effort of the user. Since the information about a domain is present on hidden web as well as on the surface web, user should explore every area of WWW to get the desired data. Thus, there is a need for a system which processes the data for the user from every corner of WWW.

Although a lot of hidden web extractors have been developed but still a very little part of hidden web has been exposed. In addition to this, the integration of processed hidden web and surface web data according to user query has not been done. User has to extract the required data manually. The problems discussed above motivated the current research. For the experimental purpose, the digital academic library has been taken in the research. The whole prototype starts with downloading web pages from WWW and then these web pages are categorized based on the type of content, such as query interfaces, research articles, call for papers, structured web pages and unstructured surface web pages. The web pages having the research articles are initially processed and clustered using the "Graph and Relationship Extraction based Clustering" method. The process is then followed by the processing of call for papers and unstructured web pages. The unstructured web pages of academic domain mostly contain the tutorials, the semantic based text mining is applied on these unstructured web pages and the content from web documents is summarized and integrated using "Semantic Based Text Mining and Summarization". Third part of the prototype is the processing of web forms. The web forms are extracted using the rule based expert system and then they are further

ranked by Web Form Extraction and Ranking module (WFER). The web forms are filled by the default value "default_X" and the generic URL templates are captured and stored. These generic URL's are then forwarded to "Expert System for Data Region Extraction and Integration of Data" module. Now modules CGRE, STMS, WEER and ESDREI are integrated and an Academic Search Engine is created at the end.

The entire research work is carried out primarily in four phases namely, Clustering using Graph and Relationship Extraction (CGRE), Semantic based Text Mining and Summarization (STMS), Web Form Extraction and Ranking of Web Forms (WFER) and Expert System for Data Region Extraction and Integration (ESDREI) as shown in Figure 4.1.

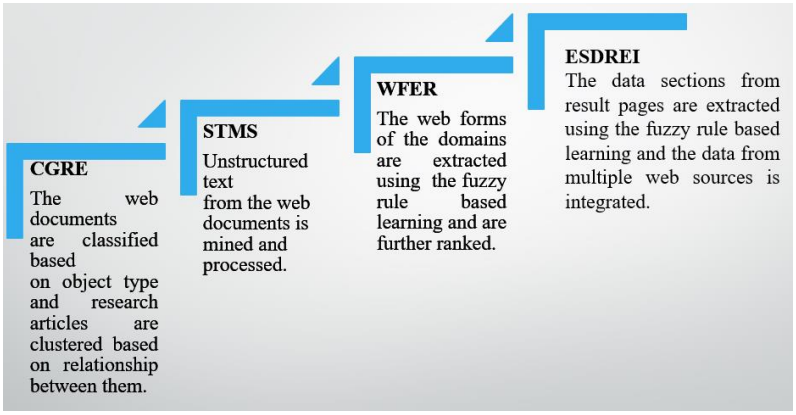

Figure 4. 1. Phases of the Proposed Work.

All the four modules classified work in parallel. After the classification of downloaded documents, they are sent to their respective object handlers and the processing is carried out independently by each module in parallel as shown in Fig 4.2.

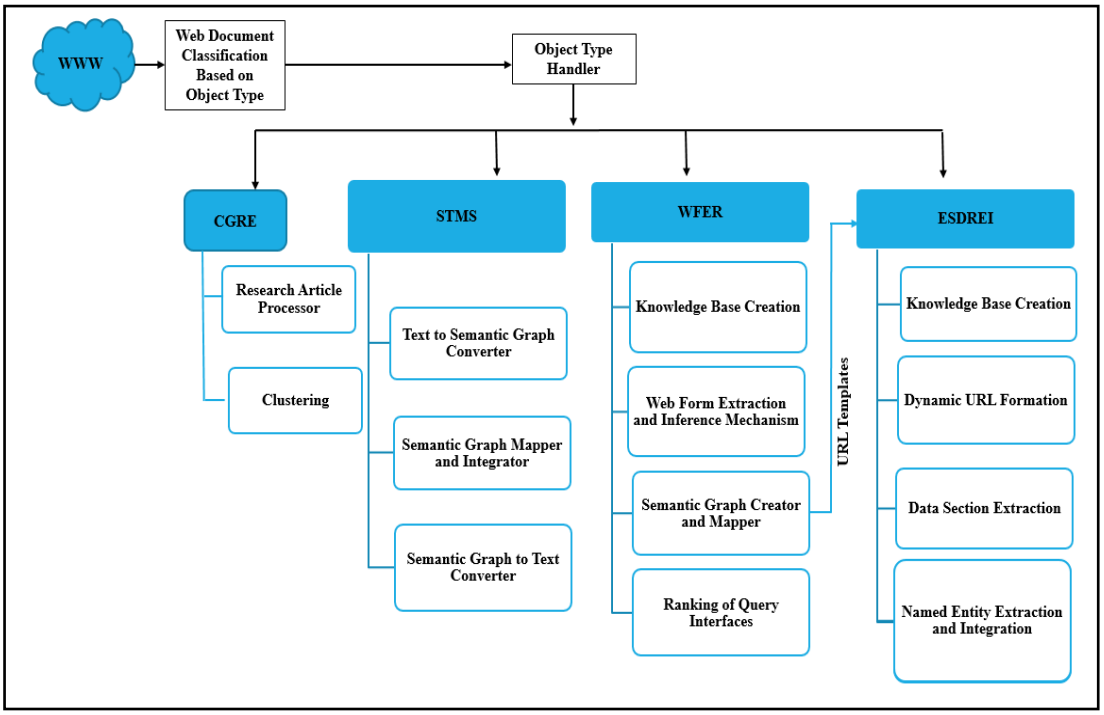

Figure 4. 2. Parallel Execution of the Proposed System Modules.

This chapter presents the details related to first two phases. The first phase "Clustering using Graph and Relationship Extraction" (CGRE) describes the processing of research articles. The second phase "Semantic Based Text Mining and Summarization" processes the unstructured web documents.

The last two phases Web Form Extraction and Ranking of Web Forms (WFER) and Expert System for Data Region Extraction and Integration (ESDREI) are discussed in Chapter 5.

4.2. DESIGN CHALLENGES

- **Heterogeneous types of web pages**

Different types of web pages contain information about the entities like unstructured data, file formats, structured data and query interfaces etc. So in order to integrate the information, all types of web pages should be processed.

- **Identification of the domains**

As there are large number of domains lying on WWW therefore in order to access the relevant information from the web pages, identification of domain is a big challenge.

- **Redundancy of Data**

  There are a lot of websites for a particular domain. So most of the information is redundant on the web. In order to extract the unique information, user has to go through all the webpages.

4.3. CLUSTERING USING THE GRAPH AND RELATIONSHIP EXTRACTION (CGRE)

The discussion presented in the previous chapters brings the fact that the content of a given domain has been scattered across different kind of web pages. When the user issue a query, the results pages can be of different types such as unstructured web pages, structured web pages, query interfaces or they can be the research articles. User has to process the web pages and extract the desired information. To overcome this limitation Clustering using Graph and Relationship Extraction (CGRE) has been proposed, which classifies the result web pages and cluster the research articles using the graph and the relationship strength between the web articles. The current section mainly clusters the research articles and shows the relevant articles to the end users.

The main aim of CGRE is to classify and cluster the web documents based on their similarity with the predefined semantic classes and the relationship between web documents. The predefined semantic classes store the basic structure of different types of web documents such as structured, unstructured, query interfaces and research articles.

Semantic Class Creation:

The data on WWW is so huge that it cannot be managed without machine intervention. In addition to this, it is very poorly rendered which makes it difficult for machines to understand. In order to make the data lying on WWW machine readable, machine-readable descriptions of the content and capabilities of Web accessible resources is required. So OWL, Web Ontology Language [123] was created to describe the classes and the relationship between the classes that are present in the web documents. The classes created using the OWL are machine readable and the documents can be processed by the machine. In the proposed work, the OWL classes are created in order to identify the type of web documents such as structured web documents, unstructured web documents, query interfaces and research articles. These OWL classes help in

identifying the web document by the help of classes and properties added to these classes.

The semantic OWL classes have been created for identifying the type of web documents. The web documents can have unstructured data, structured data, query interfaces or research articles. These types are declared using the class shown in Figure 4.3.

```
<owl: class rdf:ID=”Query Interface”>
<rdfs:subClassOf rdf:resource=”#web document”/>
</owl :class>
<owl: class rdf:ID=”Research_Article_Pdf””>
<rdfs:subClassOf rdf:resource=”#web document”/>
</owl :class>
<owl: class rdf:ID=”Structured_Document””>
<rdfs:subClassOf rdf:resource=”#web document”/>
</owl :class>
<owl: class rdf:ID=”Unstructured_Document””>
<rdfs:subClassOf rdf:resource=”#web document”/>
</owl :class>
```

Figure 4. 3. Semantic Classes Defining Types of Web Documents.

The unstructured web documents can be further classified into different types such as “call for papers” web documents, web pages having the “tutorials”. So these types are further defined by the classes as shown in Fig 4.4.

```
< owl: class rdf:ID=”Call_for_Papers”>
<rdfs:subClassOf
rdf:resource=”#Unstructured_Document”/>
</owl:class>
< owl: class rdf:ID=”Tutorials”>
<rdfs:subClassOf
rdf:resource=”#Unstructured_Document”/>
</owl :class>
```

Figure 4. 4. Semantic Classes Defining Types of Unstructured Web Documents.

Now, the classes corresponding to the different type of web documents have been defined and the respective properties are now linked with these classes which helps in identifying the type of downloaded web document. The properties for the “call for papers” types of web document are shown in Fig 4.5.

```
<owl : class rdf:Id=”Call_For_Paper”>
<owl:onProperty rdf:resource="#hasJournal_Name" />
<owl:onProperty rdf:resource="#hasArea" />
 <owl:onProperty rdf:resource="#hasLast_Date_Reg" />
<owl:onProperty rdf:resource="#hasLast_Date_Subm" />
<owl:onProperty rdf:resource="#hasDate_Result" />
<owl:onProperty rdf:resource="#hasFinal_Decision_Date" />
</owl:class>
```

Figure 4. 5. Semantic Class Definition for Call for Papers.

In the beginning, query is issued on WWW and the result pages are downloaded. Next, the downloaded web documents are classified into particular type using module “Web Document Classification based on the Object Type” as shown in Fig 4.6. For each type of classified web document, there is a unique Object Handler and these web documents are sent to their respective object handler. Each handler processes the web documents in parallel and independently. Clustering using Graph and Relationship Extraction (CGRE) is a pdf handler. Pdf files mostly contains the research articles that are clustered using the proposed technique CGRE, which uses the weighted relationship graph.

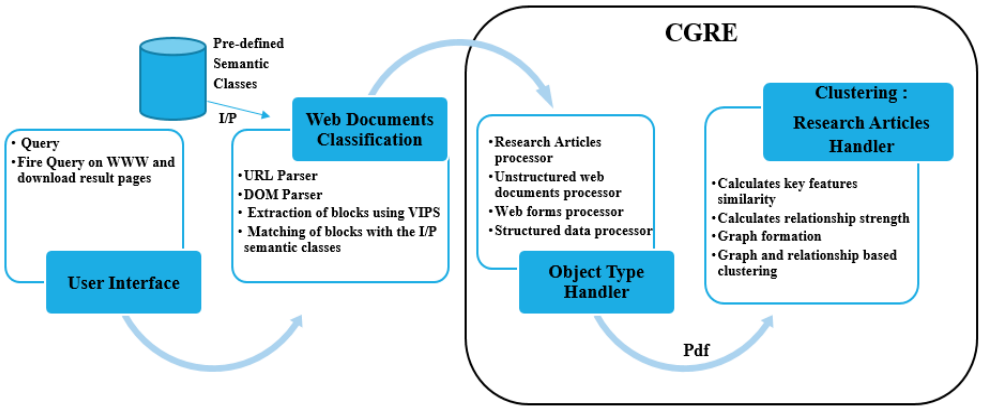

Figure 4. 6. Architecture of CGRE.

4.3.1. Web Document Classification

The downloaded web pages and semantic classes created above acts as input to web document classifier as shown in Fig 4.7. These semantic classes help in identifying whether the web page is an unstructured document or structured document or has a query interface. Firstly all pdf files get separated from other web pages using URL parser and they get stored in pdf group. After that all other the web pages get parsed using the DOM parser where all unnecessary tags or noises get removed and the web pages get segmented using the VIPS algorithm [113].

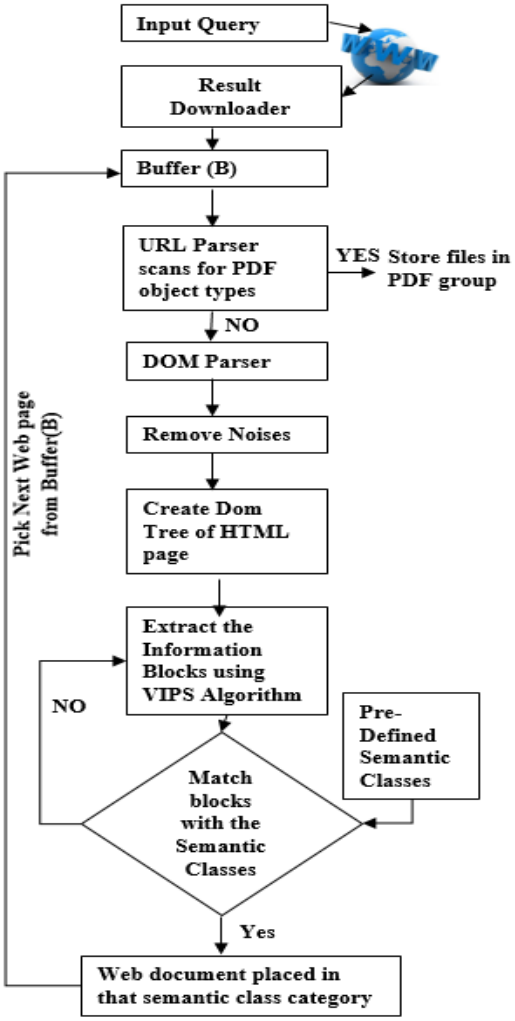

Figure 4. 7. Web Document Classifier.

The semantic blocks created using VIPS algorithm are then compared with the semantic classes and are classified based on their object types as shown in Fig 4.8. The classified web documents are sent to their respective object type handler.

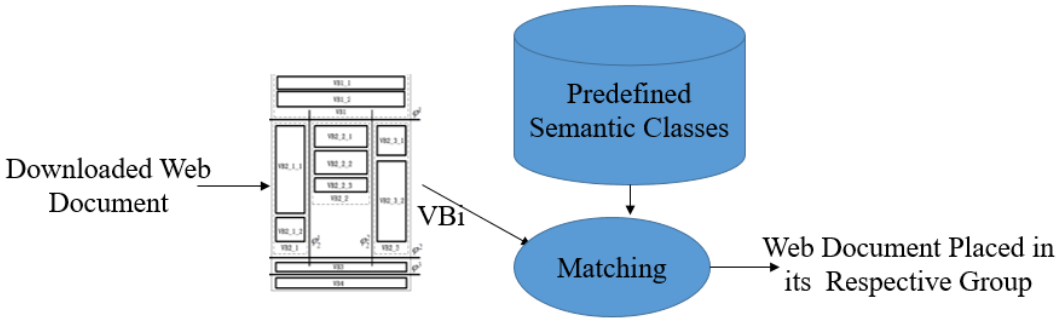

Figure 4. 8. Web Document Classification.

4.3.2. Object Type Handler

The different handlers process the classified web documents in parallel and independently as shown in Fig 4.9.

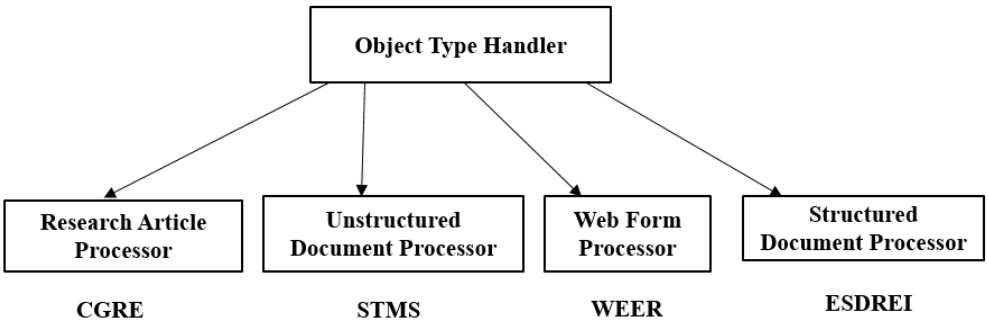

Figure 4. 9. Object Type Handler.

The research article processor works on pdf files. It clusters the articles based on significant key features, creates weighted graph of files and clusters them. Unstructured web document processor (STMS) mines, integrates and summarizes the text from multiple web sources. Web form processor (WFER) extracts the web forms from pages, generates URL templates and rank them. Structured document processor uses the rules and facts to extract the data region from the resultant pages.

4.3.3. Research Article Processor

The research article processor extracts the key feature details from the research papers and the similarity between them is calculated using the Winnowing Algorithm [113]. The research articles are then be linked to each other based on this similarity and each key feature has been assigned some weights. Now, a weighted graph is created where the different articles are connected to each other. This graph is then further sent as an input to the clustering module.

The steps of Research Article Processor are:

- Key Feature Extraction
- Key-Feature Similarity Calculation
- Weighted Graph Creation

Key Feature Extraction

There are different features with the help of which research articles can be uniquely identified and clustered. Some of the key features taken in the proposed work are given below.

- Area of research article
- Author of research article
- Journal in which research article is published
- Year in which research article is published

Key-Feature Similarity Calculation

The key features extracted are being matched using the Winnowing Algorithm [113]. Winnowing Algorithm is an extension of fingerprint algorithm. It uses window concept to overcome disadvantages of fingerprint algorithm. The fingerprints are calculated using this Winnowing algorithm. The fingerprints are then matched using the Dice Coefficient equation 1 given below.

$$Dice = \frac{2 \times \left| f(d) \cap f(di) \right|}{|f(d)| + |f(di)|} \quad (1)$$

The Dice value shows the similarity of the key feature of the two research article under processing. Similarly, the similarity of all the key features of all the research articles is calculated using the Winnowing algorithm. Now a graph is created using the values calculated above as shown in Fig 4.10.

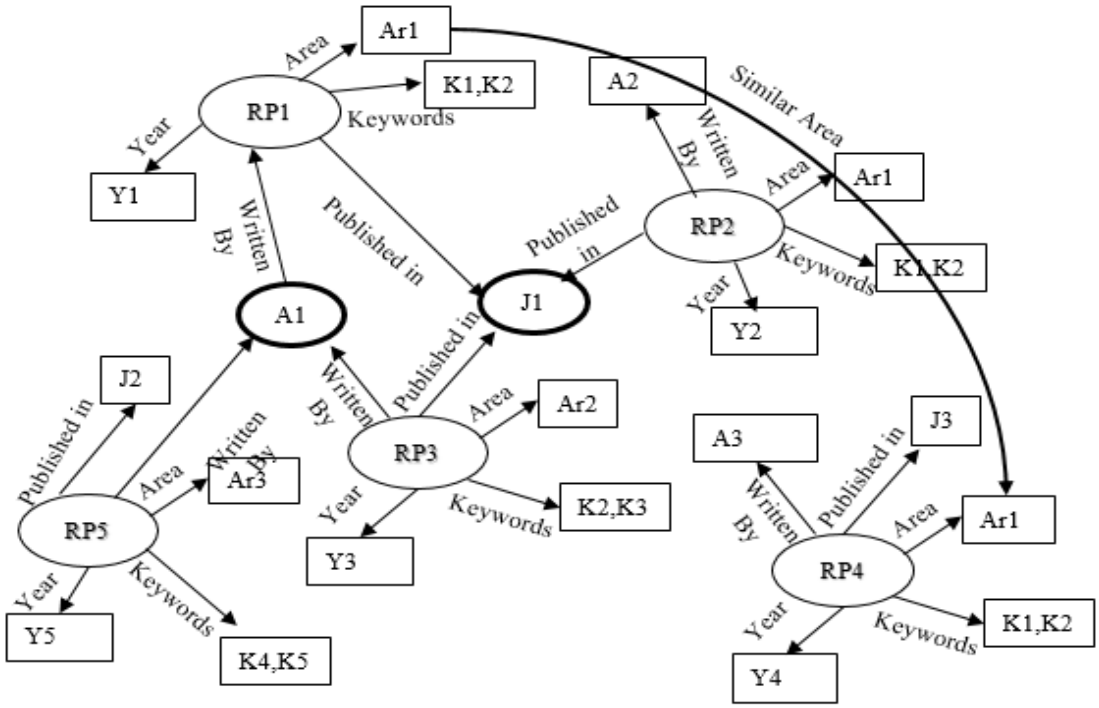

Figure 4. 10. Research Articles linked based on Key Feature Similarity.

Weighted Graph Creation

The linked graph created above is input to this step and weighted graph having research articles linked to each other is created as an output. Now the weights are assigned to the key features as shown in Table 4.1 and the total strength of link or relationship between the research articles is calculated.

Table 4. 1.Weight Assigned to Key Features.

Key Feature	Weight
Area	2
Author	1
Journal	0.35
Year	0.36

The relationship strength is calculated using the proposed formula (2) given below.

$$\text{RelSim}(\text{d}, \text{di}) = \sum_{k=0}^{m} (Dice\ (k) * weight(k) \tag{2}$$

Here d and di are the two pdf web documents, m is number of key features. Dice (k) is the similarity of the k^{th} key feature of web documents d and di. Weight (k) is the weight or the contribution of the k^{th} key feature in the relationship.

In the relationship matrix, the high value indicates that the web documents are strongly connected and the low value indicates that the web documents are loosely connected. A threshold value of "1" is set. If two documents are connected but the strength of relationship is below "1" then that relationship is discarded else the web documents are connected with edges in the graph. For the web documents processed in Fig 4.10, the relationship similarity matrix is calculated as shown in Table 4.2. The weighted relationship graph is now created for these documents as shown in Fig 4.11.

Table 4. 2. Relationship Strength Matrix.

	RA1	**RA2**	**RA3**	**RA4**	**RA5**	**RA6**	**RA7**
RA1	3.7	2.7	1.35	2.35	1	0.1	0
RA2	2.7	3.7	0.35	2.5	0	0	0.1
RA3	1.35	0.35	3.7	0.25	3	0.2	0.0
RA4	2.35	2.35	0.25	3.7	0	0	0
RA5	1	0	3	0	3.7	0	0.1
RA6	0.1	0	0.2	0	0	3.7	3.35
RA7	0	0.1	0	0	0.1	3.35	3.7

The weighted relationship graph is then sent as an input to the proposed clustering method and the relationship based clusters are created at the end.

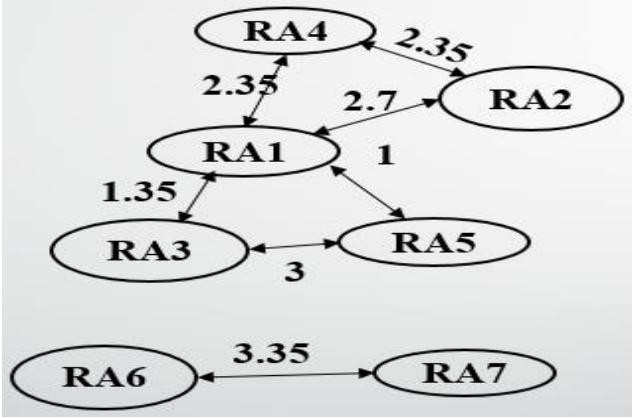

Figure 4. 11. Weighted Relationship Graph.

4.3.4. Clustering

The input to this module is the weighted graph. The weight on the edges defines the strength of the relationship between the two documents. The clustering module clusters the pdf files based on the weight and the linking of the web documents in the graph using algorithm (CGRE) as shown in Fig 4.12.

Algorithm *: Graph and Relationship Based Clustering Algorithm*
Input*: Weighted relationship graph*
Output*: Clusters (C1, C2……Cj)*
Algorithm*:*
Initially all the web documents are kept in one cluster, so number of clusters, j=1.
For i=1 to n where n is the number of web documents or nodes in the input graph
{
For the i^{th} node in the graph, extract the nodes which are directly connected to $node^i$
Place the nodes in cluster j+1
new_cluster=true;
For(k=1 to j) // Number of clusters created till now
{ If (Nodes in cluster(k) are completely similar to node in cluster(j+1))
{ discard the cluster j+1; new_cluster=false; break; }}// end of loop(k)
If(new_cluster)
{ If(few nodes from currently processed cluster j+1 found in some old cluster)
{
Repeated nodes(RN)= Extract nodes found in old cluster
Temp_OldCluster=OldCluster-RN
k=size_of(RN)
For each node in RN from 1 to k
{ if(RN(k) is not connected to any nodes from Temp_OldCluster)
{ OldCluster=OldCluster – RN(k); }
} //end of For
} // End of checking of Repeated nodes
Create a new cluster c^{j+1} and update the OldCluster and set j=j+1}
Else
{ Create a new cluster c^{j+1} from the nodes and set j=j+1 }
}
Else // Not new Cluster
{ Discard the nodes as cluster has already been made from those node
} }//end of loop(i)

Figure 4. 12. Graph and Relationship Based Clustering Algorithm.

In the beginning all the nodes from the weighted relationship graph shown in Fig 4.11 are placed in one cluster. So the first cluster C1 have all the 7 nodes from the weighted relationship graph.

C1= (RA1, RA2, RA3, RA4, RA5, RA6, RA7)

Now the nodes from the weighted relationship graph are picked one by one. For example the node RA1 is picked initially. It is then followed by selecting the adjacent nodes and new nodes are now placed in new cluster i.e. C2 cluster. So the nodes in the cluster C2 are.

C2= (RA1, RA2, RA3, RA4, RA5)

Now newly created cluster i.e. C2 is compared with previously created cluster. The first comparison is whether the newly created node is exactly similar to the previously created nodes or not. For example the newly created cluster is C2 and the previously created cluster is C1 here and it has been found that newly created cluster C2 is not similar to old cluster C1. The nodes from newly created cluster and the old clusters have now been checked for overlapping. If overlapping is not there, then the new cluster will be set else overlapped nodes will be processed. Here C1 and C2 have overlapped nodes (RA1, RA2, RA3, RA4, and RA5) as they are present in both the clusters. So the overlapped nodes (RA1, RA2, RA3, RA4, and RA5) are processed in next step. The temporary old cluster will be created from which overlapped or repeated nodes from old cluster will be removed. The steps for processing the repeated/overlapped nodes are given below.

- Temp_old_cluster=Old_Cluster – RN
  So Temp_old_cluster=(RA6, RA7)
- RN= Overlapped Nodes (RA1, RA2, RA3, RA4, RA5)

Each node from RN will be picked and it will be checked whether it is connected to any node from Temp_old_cluster. If yes then that node will be retained in old cluster else that node will be removed from old cluster. In the current case, node RA1 is not connected to any node from Temp_old_cluster. So RA1 will be removed from old cluster C1. This will be repeated for every node from RN set. None of the nodes from RN here are connected to Temp_old_cluster nodes. So all the nodes from RN will be

removed from old cluster C1. Finally C1 will be updated to having 2 nodes RA6, RA7. After first Iteration two clusters are created C1 and C2.

C1= (RA6, RA7), C2= (RA1, RA2, RA3, RA4, RA5).

The same process has been repeated for second node RA2 and the resultant clusters are.

C1= (RA6, RA7), C2= (RA1, RA2, RA4) and C3= (RA1, RA3, RA5)

Let's take node RA3, its adjacent nodes are extracted and are placed in cluster C4= (RA1, RA3, RA5). C4 cluster is identical to cluster C3. So cluster C4 is not created. The same process has been repeated for all the remaining nodes RA4, RA5, RA6, RA7 and the finally the clusters are.

C1= (RA6, RA7), C2= (RA1, RA2, RA4) and C3= (RA1, RA3, RA5)

The clusters created are shown in Fig 4.13.

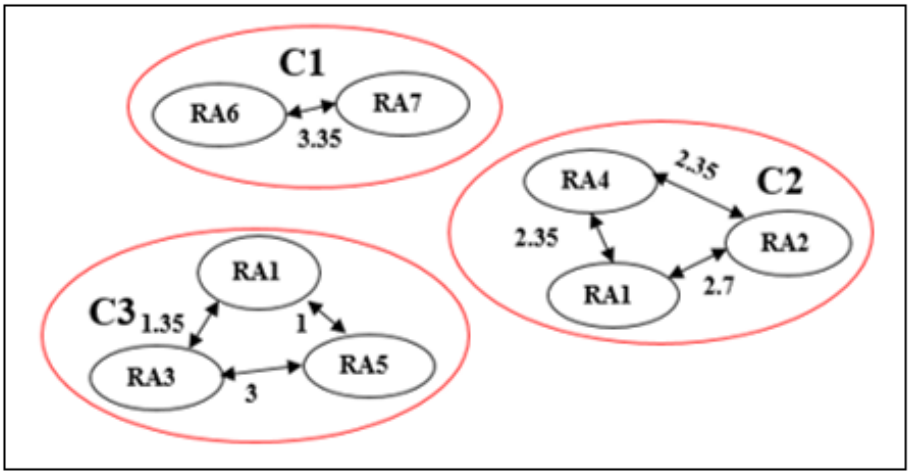

Figure 4. 13. Output Clusters.

4.4. SEMANTIC BASED TEXT MINING AND SUMMARIZATION

Semantic based Text Mining and Summarization (STMS) proposes an algorithm Semantic Graph Mapper and Integrator (SGMI) that integrates and summarizes the text from multiple web documents using semantic networks. Let's consider the scenario where a user wants to learn about a topic such as "Internet of Things". The end user issues the query and he gets a set of result pages from WWW. It is often found that most of the information is redundant and to extract the unique information he/she has to filter out content from multiple web pages. The proposed technique STMS removes the redundant information lying on multiple web pages and shows integrated and

summarized results to the user. When a query is submitted to the search engine, the proposed system extracts the information lying on multiple web pages as shown in Fig 4.14 and algorithm SGMI is shown in Fig 4.19.

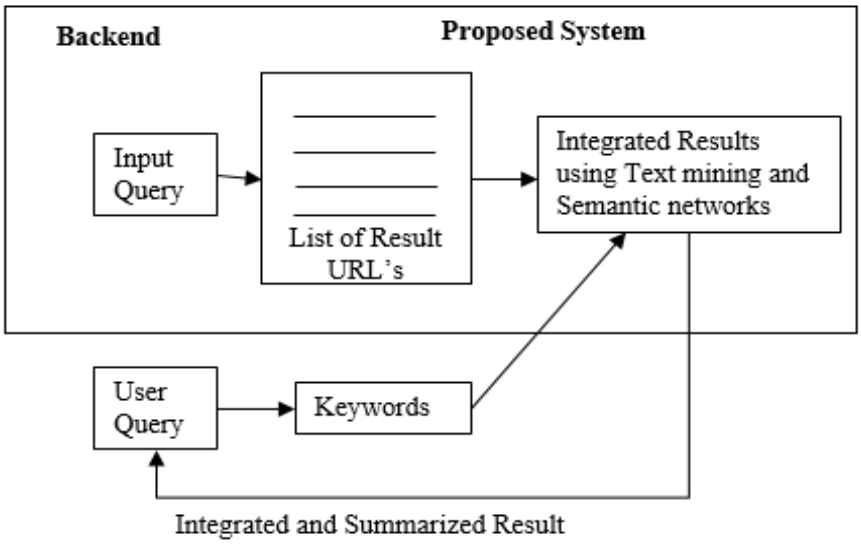

Figure 4. 14. Proposed STMS Architecture.

The system starts with creating semantic network or graph from the text found in the web documents. In the semantic network, the sentences get stored along with their meaning. If two sentences are there in the different web documents which have different keywords but are semantically similar then they are mapped in the semantic graph and the sentence occurs only once in the integrated result. The sentences that have been found frequently on most of the web pages are ranked based on their frequency and hence the summarized and integrated results are shown to the user.

The proposed architecture has three modules namely Text to Semantic Graph Converter (TSGC), Semantic Graph Mapper and Integrator (SGMI) and Semantic Graph to Text Converter (SGTC) as shown in Fig 4.15.

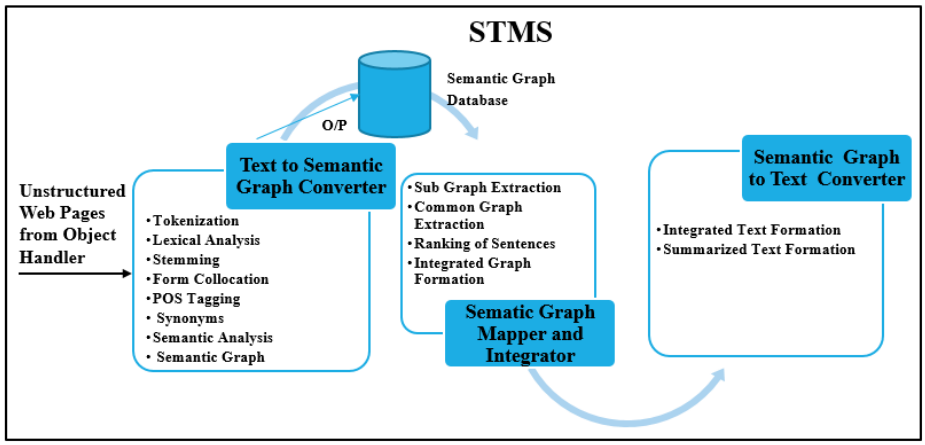

Figure 4. 15. Architecture of STMS.

The Text to Semantic Graph Converter creates a semantic graph from the sentences found in the web document. Semantic Graph Mapper and Integrator maps the semantic graphs of different web documents and removes the duplicate sentences. While mapping the sentences, the sentences which are found in multiple web documents, are ranked too. The output of SGMI is an integrated semantic graph or network and the summarized result which is generated by selecting the highly ranked sentences. In the end when the user will issue a query, the integrated and the summarized result are shown to the user.

4.4.1. Text to Semantic Graph Converter

The WWW is filled with a lot of unstructured text. The computers are not able to understand this unstructured text. In order to make text machine understandable or to process it, it has to be converted into some structured format. TSGC converts the unstructured text from multiple web pages into a graph structure. The input to TSGC is the text from multiple web documents and the output is the semantic graph, which gets stored in semantic graph database. The nodes in the graph describe noun, object, verb, action, event etc. while the edges describe the meaning or the relationship between the nodes. TSGC is done in two steps namely text pre-processing and semantic graph creation as shown in Fig 4.16.

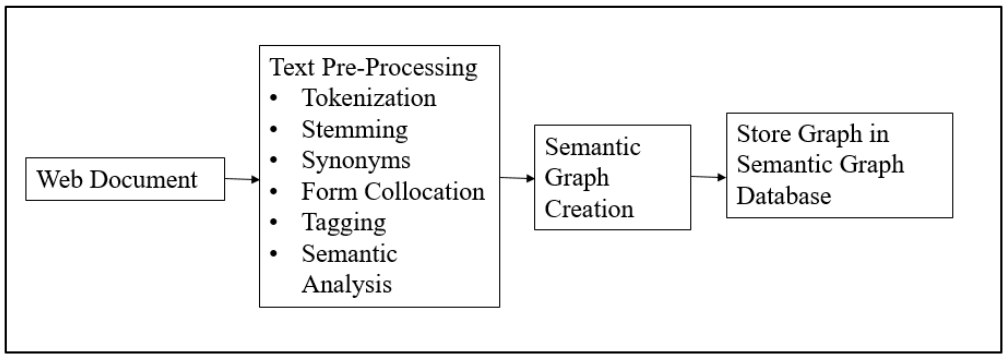

Figure 4. 16. Text to Semantic Graph Converter.

Text- Pre-processing:

Text-pre-processing prepares the document for further processing. Text-pre-processing is carried out using the steps given below.

- Tokenization: Tokenization is also known as lexical analysis. In tokenization the text from the unstructured document is taken as input and stream of words is generated as the output.
- Stemming: In English there are different forms of a single word. For example the word “run” has different forms such as running, ran, runs etc. In stemming all these different forms are converted into one. So running, ran, runs are converted to run.
- Synonym Finding: All the synonyms of a word are found and are replaced by a single word.
- Form Collocation: In English there are certain words which are used together such as source code and target code. These two words have the same meaning. In form collocation such words are identified and are replaced.
- Tagging: There are different words in a sentence. Part of speech (POS) helps to identify why a word is used in the sentence. The word in a sentence can be noun, pronoun, adjective, verb, adverb, preposition etc. Tagging is used to identify whether a word is noun, verb etc.
- Semantic Analysis: Semantic Analysis helps in identifying the meaning of the sentence.

Semantic Graph Creation:

In this sub module the tagged tokens and the relationships between them are used as the input and the semantic graph is generated as output. The sequence number and the frequency of occurrence of the sentence get stored in the graph.

The above steps are repeated for every sentence in the web document and the sentences are be linked to each other in the graph. The complete semantic graph is created for the web document and gets stored in semantic graph database. The TSGC algorithm has been shown in Fig 4.17.

***Algorithm**: Text to Semantic Graph Converter*

***Input**: Web document Set (WD1……WDn)*

***Output**: Semantic Graph Database*

Algorithm:

1. *For(i=1 to n) // n is total number of web documents in the set*

{

2. *For each sentence in web document*

{

2.1. Pre-process the sentence

{

Extract the Tokens from the sentence

Stemming of words found in sentence

Finds the synonyms and store them

Tagging of token as Agent, Event or objects

Form collocation of words

Semantic Analysis of the words

}

2.2. Create a semantic graph of tagged tokens, store the location of the sentence and assign occurrence frequency =1 to the current processed sentence.

2.3. Link the sentence to the existing graph of the current web document.

}

3. *Store the semantic graph of the web document in the semantic Graph Database. }*

Figure 4. 17. Text to Semantic Graph Converter Algorithm.

4.4.2. Semantic Graph Mapper and Integrator

This module takes semantic graph database as input that results into summarized and integrated graph as shown in Fig 4.18. The results of a query contain the unstructured web documents where most of the information is redundant. This module extracts the unique content from multiple web documents, ranks the sentences based on frequency of occurrence and summarizes the topic based on rank of sentences as shown in Fig 4.19.

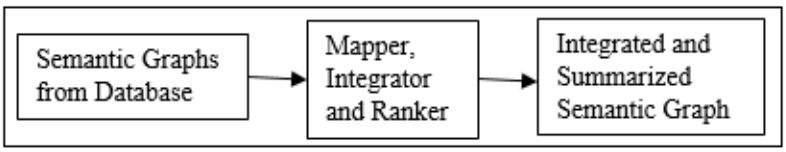

Figure 4. 18. Semantic Graph Mapper and Integrator.

Input: *Semantic Graph Database*

Output: *Integrated and Summarized Semantic Graph*

Algorithm:

1. *Integrated Graph (IG)= First Graph from the SGD*
2. *For i=2 to n (Total number of networks in the SGD)*

   { 2.1. Map the synonyms in IG and SGD ^{i}

   2.2. Sub-Graph = difference between the IG and SGD ^{i}

   2.3. Common graph= IG intersection SGD ^{i}

   2.4. Increment the frequency of sentences in the common graph

   2.5. IG= Merge the sub-graph and common graph

   }
3. *Extract the frequently occurring sentences from the IG and create the summary*

Figure 4. 19. Semantic Graph Mapper and Integrator Algorithm.

4.4.3. Semantic Graph to Text Converter

The reverse engineering has been done in this module by converting the integrated graph created in second step back to text. The sequence and frequency of occurrence of the sentences have already been stored in the integrated semantic graph. The sentences are picked sequentially and the nodes in the graph have been traversed to formulate sentences. The sentence which occurs frequently and it is on top of each page will be extracted to create the summary.

Illustration:

The web documents are downloaded using the Google API. To test the system, 100 queries are fired. The sample of input queries are:

- Human Computer Interaction
- Compiler Optimization
- Parallel Computing
- Social Networks
- Internet of Things
- Hidden Web
- Big Data
- Java Introduction

The query is fired and 100 result pages are downloaded for each query. Let's consider the query "Java Introduction" is fired and set of result pages obtained from the search engine are shown in Fig 4.20.

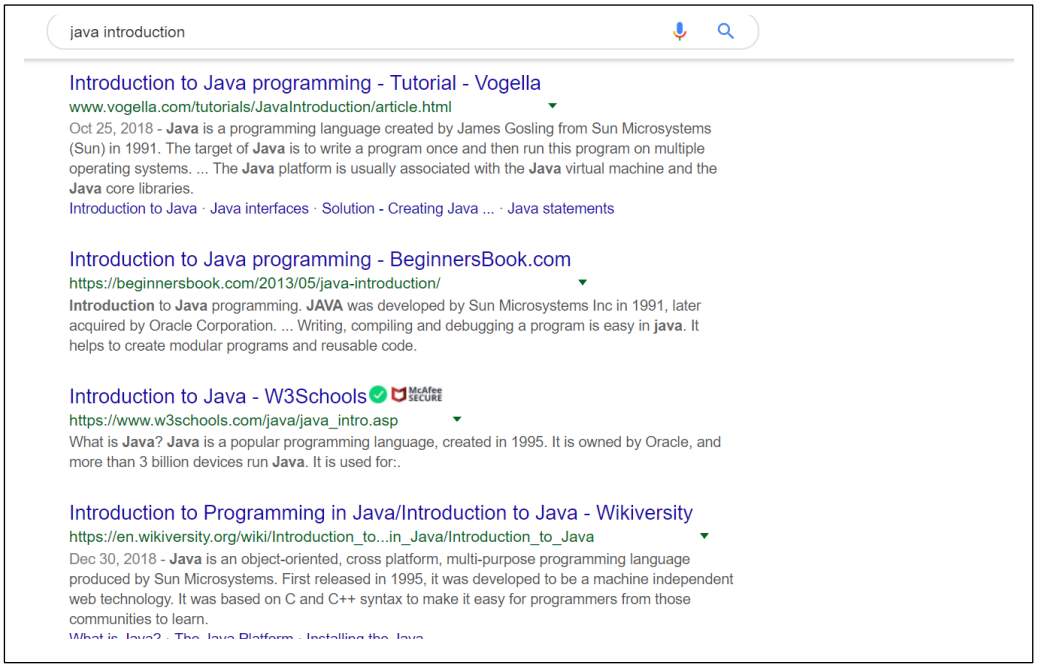

Figure 4. 20. Results Returned by Search Engines.

The text is extracted from each link and the document set, "WD1…….WDn" is created. A semantic graph is created for each document. For example from the URL "https://www.w3schools.in/java-tutorial/intro/" the text is extracted and is placed in

WD1. The few lines from the document are in Fig 4.21. Each sentence is picked one by one and the semantic graph is created from the text as shown in Figure 4.22.

"Java is an object-oriented programming language developed by Sun Microsystems and released in 1995. Java was originally developed by James Gosling at Sun Microsystems (which has since merged into Oracle Corporation). Java programs are platform independent which means they can be run on any operating system with any type of processor as long as the Java interpreter is available on that system. Java code that runs on one platform does not need to be recompiled to run on another platform, it's called write once, run anywhere (WORA). "

Figure 4. 21. Sample Text from First Web Document.

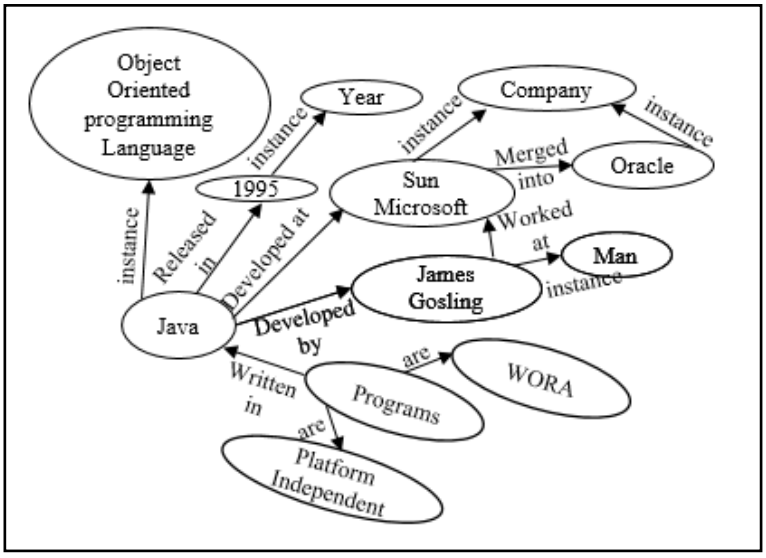

Figure 4. 22. Semantic Graph for WD1.

From the second URL "https://beginnersbook.com/2013/05/java-introduction/" the text is extracted and is placed in WD2 as shown in Fig 4.23. The semantic graph is created from the text as shown in Figure 4.24.

"JAVA was developed by Sun Microsystems Inc in 1991, later acquired by Oracle Corporation. It was developed by James Gosling and Patrick Naughton. It is a simple programming language. Writing, compiling and debugging a program is easy in java. It helps to create modular programs and reusable code."

Figure 4. 23. Sample Text from Second Web Document.

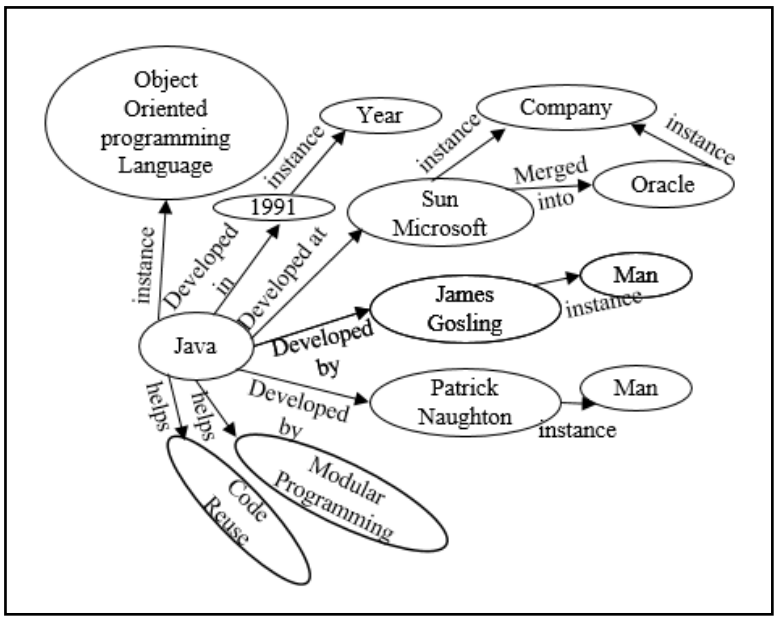

Figure 4. 24. Semantic Graph for WD2.

Now the semantic graphs are compared, unique and common contents are extracted from the graphs of WD1 and WD2 as shown in Fig 4.25 and Fig 4.26.

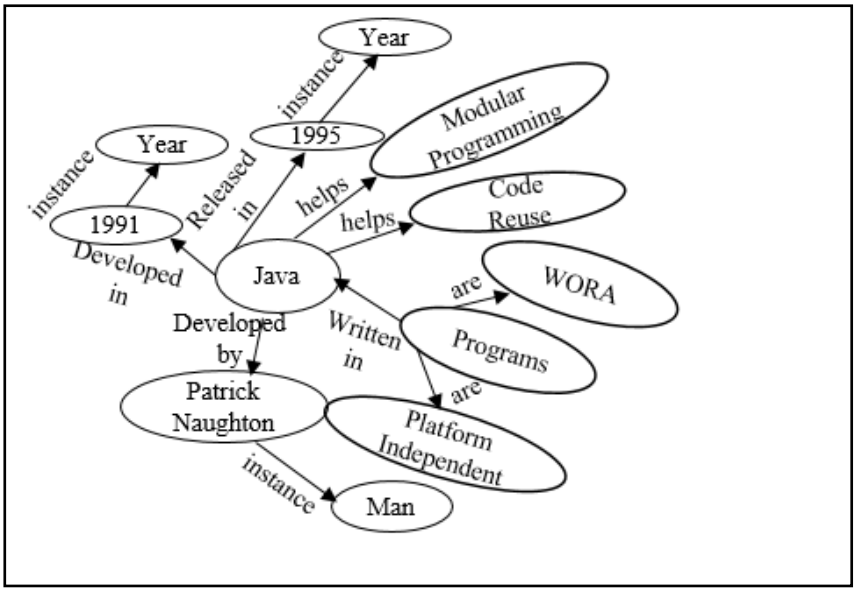

Figure 4. 25. Unique Content from Semantic Graphs of WD1 and WD2.

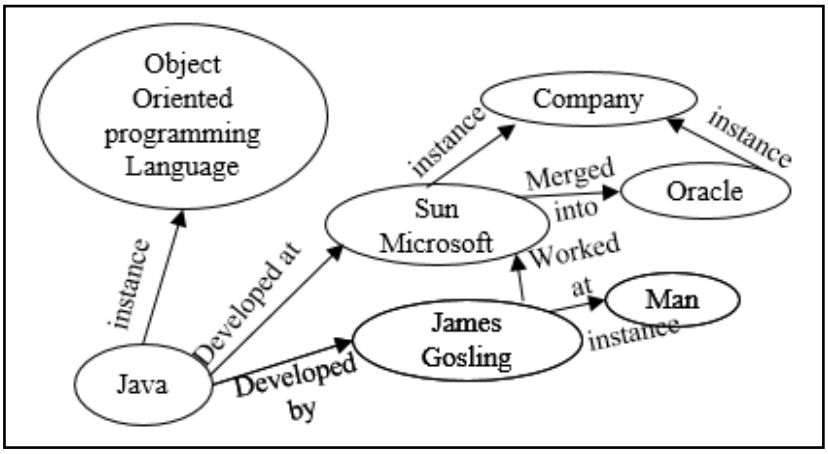

Figure 4. 26. Common Content from Semantic Graphs of WD1 and WD2.

Now, the integrated graph is created by merging the unique and common graphs as shown in Fig 4.27. The integrated graph stores both the sequence and the frequency of the sentences like n1, n2 etc. as shown in Fig 4.27.

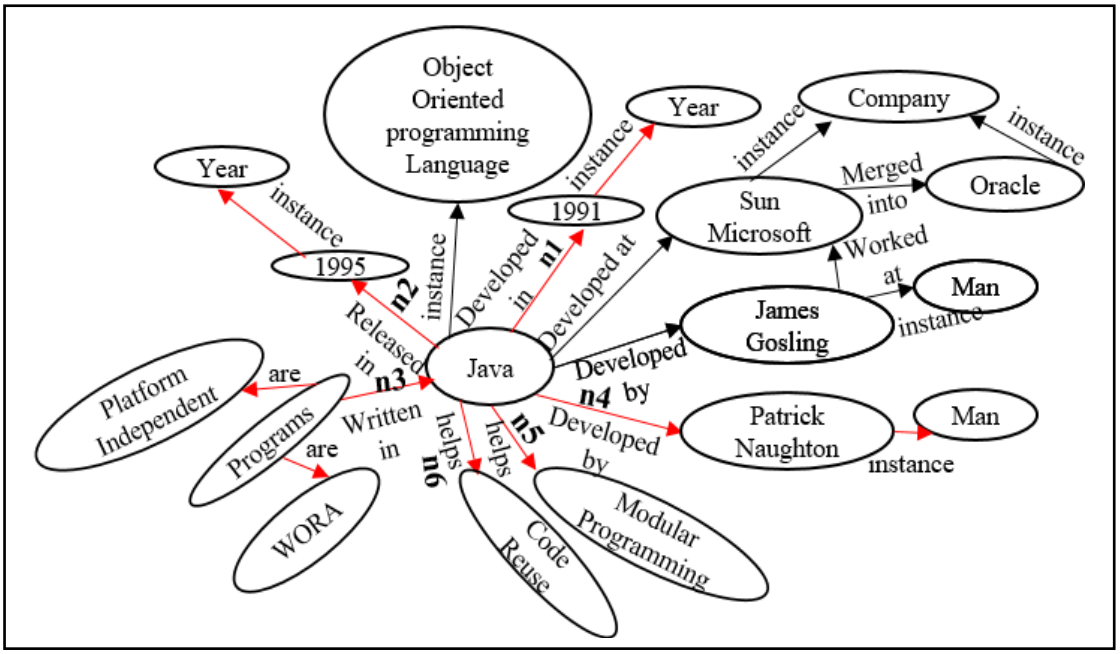

Figure 4. 27. The Integrated Semantic Graph of WD1 and WD2.

The integrated sematic graph is now converted back to text by using the graph traversal and the sequence of sentences stored in the graph as shown in Fig 4.28.

"Java is an instance of object oriented programming language. Java was developed in 1991.Java was released in 1995. Java was developed at sun Microsoft Company. Microsoft Company is merged into Oracle now. Java was developed by James Gosling and Patrick Naughton. James Gosling worked at Sun Microsoft. Java helps in Modular Programming and code reuse. Java programs are platform independent and WORA."

Figure 4. 28. Integrated Text from Multiple Web Documents.

The frequency of sentences have been analysed and the highly frequent sentences along with their sequence of occurrence are extracted. The nodes involved in those sentences are traversed and the summary of the text is generated as shown in Fig 4.29.

Java is an instance of object oriented programming language. Java was developed at sun Microsoft Company. Microsoft Company is merged into Oracle now. Java was developed by James Gosling .James Gosling worked at Sun Microsoft.

Figure 4. 29.Summarized Text from Multiple Web Documents.

4.5. CONCLUSION

This chapter presents the details about the first two modules of the proposed system namely, CGRE and STMS. CGRE mainly processes the research articles while STMS processes the unstructured web documents. CGRE has three main steps. In first step, the downloaded web documents are classified as structured web documents, unstructured web documents, query interfaces and research articles. In the second step, the web documents are passed to their respective handlers. In the last step, the clustering of the research article is done using the proposed graph and relationship based clustering technique. In the second module STMS, the unstructured web documents containing the tutorials are processed, summarized and integrated. STMS has three sub modules namely Text to Semantic Graph Converter, Semantic Graph Mapper and Integrator and Semantic Graph to Text Converter. The third and the fourth module of the proposed work, namely WFER and ESDREI will be discussed in Chapter 5.

CHAPTER 5

EXPERT SYSTEM FOR WEB FORM EXTRACTION AND HIDDEN DATA EXTRACTION

5.1. INTRODUCTION

As already discussed, the proposed work comprises of four main phases CGRE, STMS, WFER and ESDREI, where each phase has certain role and functionality in the design of Dynamic Query Processing for Hidden Web Data Extraction (DQPHDE). In chapter 4 the first two phases CGRE and STMS have been discussed. CGRE clusters the research articles and classify the different types of web documents. The second component STMS summarizes and integrates data from unstructured web pages. The remaining two components Web Form Extraction and Ranking (WFER) and Expert System for Data Region Extraction and Integration (ESDREI) are discussed in this chapter. Both the proposed components WFER and ESDREI exploits the fuzzy rule based expert system for the understanding of domain, query interfaces of the domain and the data containing sections of the result web pages.

WFER extracts the web forms from the web pages, using the fuzzy rule based expert system, generates the generic URL templates and ranks them using the inlinks, outlinks and the benefit of links. WFER has four sub modules namely knowledge base creation, web form extraction and inference mechanism, semantic graph creator and mapper and ranking of web forms. ESDREI has four sub modules namely knowledge base creation, URL formation, data section extractor, named entity extractor and integrator. ESDREI takes structured web documents as input and generic URL templates get completed with user query and structured result web pages are downloaded. It uses the fuzzy rule based system to extract the data region from the structured result web pages, processes data from them removes redundancy and integrates data from multiple web sources.

5.2. Web Form Extraction and Ranking (WFER)

Web Form Extraction and Ranking (WFER) is a fuzzy rule based expert system that has a knowledge base composed of rules and facts, which helps in identification of the domain and the web forms of that domain. The traditional search engines are not able

to process the web forms by their own. They need the human intervention to identify the web forms of the domain and extract the data behind the web forms as shown in Fig 5.1.

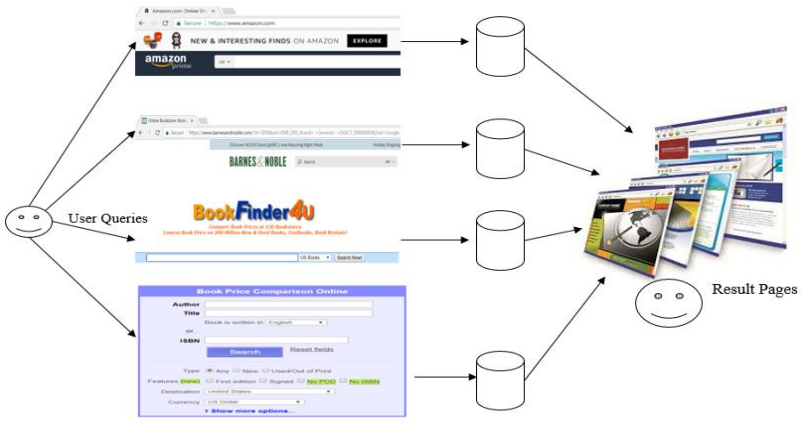

Figure 5. 1. Interaction of End Users with the Web Forms

The proposed system WFER uses the intelligence of the human expert and acts like end users. WFER identifies the web forms, rank them and extract the data behind the web forms without the intervention of end users. The success of any expert system depends upon the knowledge owned by it. The four submodules of WFER are shown in Fig 5.2.

- Knowledge Base Creation
- Web Form Extraction and Inference Mechanism
- Semantic Graph Creator and Mapper
- Ranking of Web Forms

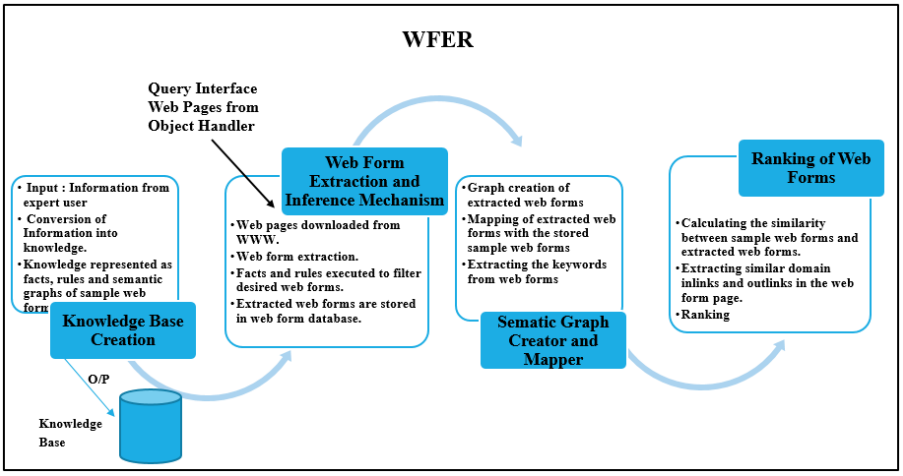

Figure 5. 2. Modules of Web Form Extraction and Ranking of Web Forms

In knowledge base creation the information about the domain is gathered which is then processed. The rules and the facts are generated from the gathered information. In the next step, the web pages of the desired domain are downloaded by using the Google API. From the downloaded web pages, the web forms are extracted by Web Form Extraction module and get stored in the web form database. The web forms under processing are placed in the working memory. The web form from the working memory and the rules and facts from knowledge base are then sent to the inference mechanism. The inference mechanism uses the facts and rules and decides whether the web form is of the desired domain or not. When the web form is matched using the rules and facts, then it get checked for whether it is a single search box web form or multiple search box web form. In the next step the semantic graph of the selected web form is generated. In the last step, the rank of the query interface is calculated using the similarity of web forms with the sample web forms and the number of in-links and out-links present in the web page. The domain of the inlinks and outlinks found should be same as that of the query interface under processing. For example “foboko.com” has inlink from “free-ebboks.net” which is of same domain book and similarly “acadzone.com” has outlink to “bookadda.com” which is also of same domain book.

5.2.1. Knowledge Base Creation

The end users while processing the web forms are able to show their intelligence only when they have some previous knowledge or experience about the domain. Similarly, in order to make the system intelligent in a domain, the knowledge has to be added into the system. The input to this module is the domain information and the high-quality information about the sample web forms as shown in Fig 5.3.

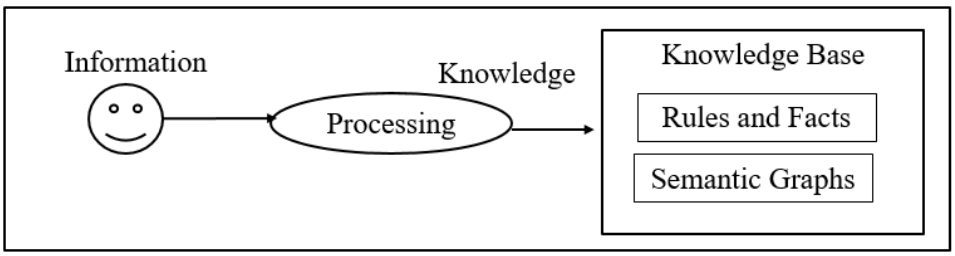

Figure 5. 3. Knowledge Base Creation

This information is now converted into rules, facts and semantic entity graph. By the help of these rules, facts and semantic entity graphs, the web form graphs are created. The facts help in describing the domain and the rules help in finding the web forms of

the desired domains automatically. The algorithm for knowledge base creation is shown in Fig 5.4.

Algorithm: *Knowledge base creation*

Input: *Domain knowledge and sample web forms details*

Output: *Semantic graph of domain, sample web form's semantic graph, rules and facts*

1. Create the rules and fact.

2. Create semantic entity graph.

3. Add sample web forms as web form graphs.

Figure 5. 4. Knowledge Base Creation Algorithm

For the experimental purpose, the academic domain is taken. When a user issues the query on WWW related to any topic related to academics then the result pages have details such as tutorials, research articles, books etc. The high-quality information about the domain is provided. For example book has title, price, volume, publication, author, and subject. So this knowledge is added into the system as a semantic entity graph, with the meaning and data type of each information as shown in Fig. 5.5.

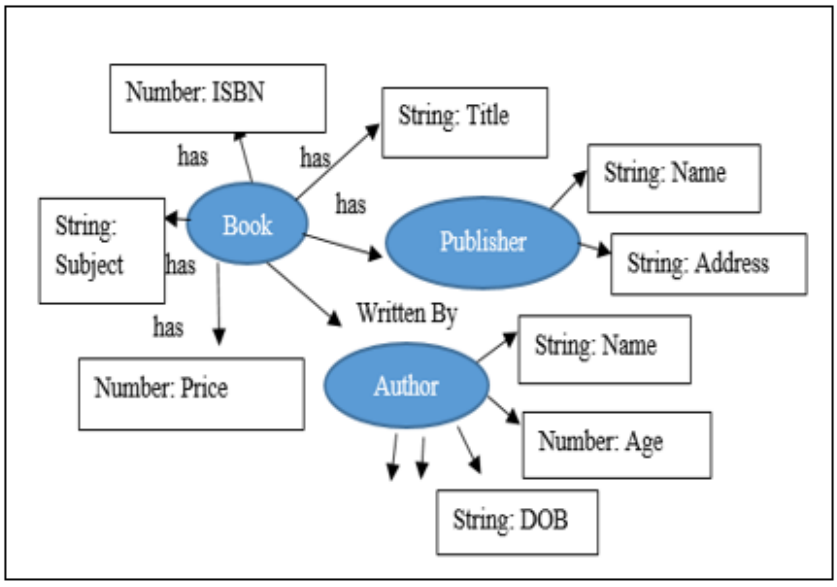

Figure 5. 5. Semantic Graph of Book Domain.

In the next step, the details of few sample query web forms are added. The sample query web forms are categorized as single search box web forms (SSBWF) and multiple

search box web forms (MSBWF). The basic structure of both these web forms now get stored as semantic web form graphs as shown in Fig. 5.6.

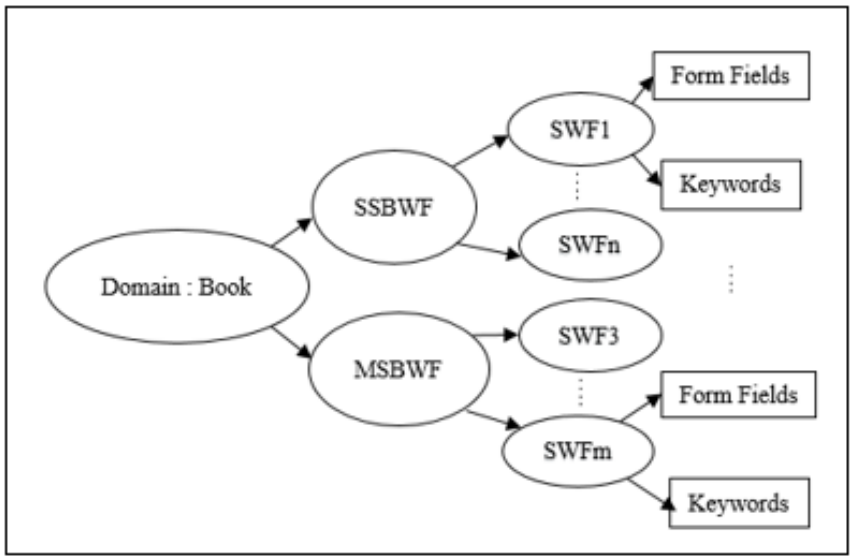

Figure 5. 6. Web Form Graph.

For example the keywords found in the sample forms along with the frequency of occurrence of each keyword get stored in the web form graph. The remaining components of the knowledge base are the rules and facts that are created from the domain knowledge and the web forms details provided to system. For example to describe the “book” domain the set of facts are shown in Table 5.1.

Table 5. 1. Facts for Book Domain.

Fact_ID	Fact
F1	Written_by (book, author)
F2	Has (book, name)
F3	Has (book, title)
F4	Has (book, subject)
F5	Published (book, year)
F6	Has (book, cost)
F7	Has (book, volume)
F8	Has_unique (book, ISBN)

The web form of Single Search Box Web Form (SSBWF) is shown in Fig 5.7(a). The rules for the SSBWF of book domain are listed in Table 5.2. These rules are used for selecting the web forms which have single search box.

Figure 5. 7. Sample Web Forms.

Table 5. 2. Rules for Single Search Box Web Forms.

Rule_ID	**Rule**
R1	Has(WF, textfield) ^ Has(WF, Submit Button) ^contains(WF, Enter Text here)→ Selected(WF, SSBWF, Book)
R2	Has(WF, textfield) ^ Has(WF, Submit Button) ^ contains (WF, Enter name of Book Here)→ Selected(WF, SSBWF, Book)
R3	Has(WF, textfield) ^ Has(WF, Submit Button) ^ contains (WF, Search for a book))→ Selected(WF, SSBWF, Book)
R4	Has(WF, textfield) ^ Has(WF, Submit Button) ^ contains(WF, Book name))→ Selected(WF, SSBWF, Book)
R5	Has(WF, textfield) ^ Has(WF, Submit Button) ^ contains(WF, search)→ Selected(WF, SSBWF, Book)
R6	Has(WF, textfield) ^ Has(WF, Submit Button) ^contains(WF, Enter Text here)→ Selected(WF, SSBWF, Book)

The web forms of Multiple Search Box Web Forms (MSBWF) are shown in Fig 5.7(b) and Fig 5.7(c). The rules for the MSBWF of book domain are listed in Table 5.3. These rules are used for selecting the web forms which have multiple search box.

Table 5. 3. Rules for Multiple Search Box Web Forms

Rule_ID	**Rule**
R7	Has(WF, textfield) ^ Has(WF, Radio Buttons) ^ Has(WF, Submit Button) ^ (contains(WF, Title)^contains(WF, Subject)→ Selected(WF, MSBWF, Book)
R8	Has(WF, textfield) ^ Has(WF, Radio Buttons) ^ Has(WF, Submit Button) ^ (contains(WF, Title) ^ contains(WF, Author))→ Selected(WF, MSBWF, Book)
R9	Has(WF, textfield) ^ Has(WF, Radio Buttons) ^ Has(WF, Submit Button) ^ (contains(WF, Title) ^ contains(WF, ISBN))→ Selected(WF, MSBWF, Book)
R10	Has(WF, textfield) ^ Has(WF, Radio Buttons) ^ Has(WF, Submit Button) ^(contains(WF,Title)^contains(WF, Publisher))→ Selected(WF, MSBWF, Book)
R11	Has(WF, textfield) ^ Has(WF, Radio Buttons) ^ Has(WF, Submit Button) ^(contains(WF, Author) ^ contains(WF, Publisher))→ Selected(WF, MSBWF, Book)
R12	Has(WF, textfield) ^ Has(WF, Radio Buttons) ^ Has(WF, Submit Button) ^(contains(WF,Author) ^ contains(WF, ISBN))→ Selected(WF, MSBWF, Book)
R13	Has(WF, textfield) ^ Has(WF, Radio Buttons) ^ Has(WF, Submit Button) ^(contains(WF, Author) ^ contains(WF, Subject))→ Selected(WF, MSBWF, Book)
R14	Has(WF, textfield) ^ Has(WF, Radio Buttons) ^ Has(WF, Submit Button) ^(contains(WF, Publisher) ^ contains(WF, ISBN))→ Selected(WF, MSBWF, Book)
R15	Has(WF, multiple text field) ^ Has(WF, Submit Button) ^ (contains(WF, Title)^contains(WF, Subject)→ Selected(WF, MSBWF, Book)
R16	Has(WF, multiple text field) ^ Has(WF, Submit Button) ^ (contains(WF, Title) ^ contains(WF, Author))→ Selected(WF, MSBWF, Book)
R17	Has(WF, multiple text field) ^ Has(WF, Submit Button) ^ (contains(WF, Title) ^ contains(WF, ISBN))→ Selected(WF, MSBWF, Book)
R18	Has(WF,multiple text field)^ Has(WF,Submit Button)^ (contains(WF,Title)^contains(WF, Publisher))→ Selected(WF, MSBWF, Book)
R19	Has(WF, multiple text field) ^ Has(WF, Submit Button) ^(contains(WF, Author) ^ contains(WF, Publisher))→ Selected(WF, MSBWF, Book)
R20	Has(WF, multiple text field) ^ Has(WF, Submit Button) ^(contains(WF,Author) ^ contains(WF, ISBN))→ Selected(WF, MSBWF, Book)
R21	Has(WF, multiple text field) ^ Has(WF, Submit Button) ^(contains(WF, Author) ^ contains(WF, Subject))→ Selected(WF, MSBWF, Book)

5.2.2. Web Form Extraction and Inference Mechanism

The input to this module is the knowledge base and the downloaded web pages from the WWW. The output of the module is the collection of query interfaces of the required domain. When query is fired and the result web pages are downloaded from WWW, the downloaded web pages may have both the searchable web forms and the non-searchable web forms. First of all, the non-searchable web forms are removed. The login and signup forms are considered to be non-searchable web pages, these pages are also eliminated. The rest of the web forms are now processed by matching with the sample web forms using the inference mechanism as shown in Fig 5.8.

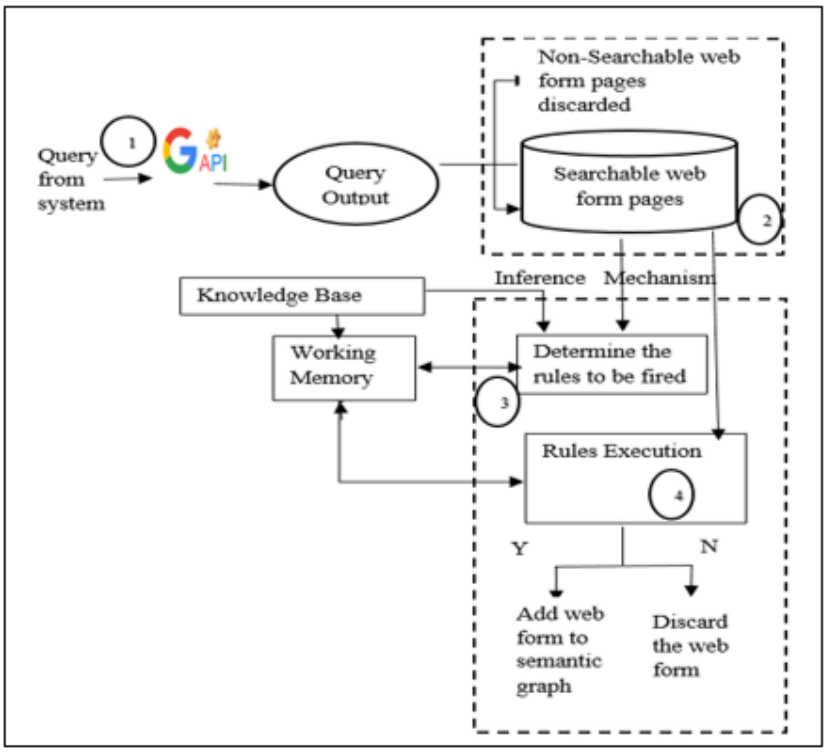

Figure 5. 8. Web Form Extraction and Inference Mechanism.

The inference mechanism makes use of the facts, rules, web form graphs and the semantic graph of the domain. If the query interface or web form matches with any of the rules from the knowledge base, then that web page get stored in temporary memory for further processing. The algorithm for web form extraction and inference mechanism is discussed in Fig 5.9.

Algorithm: *Web form Extraction and Inference Mechanism*

Input: *The knowledge base and web pages downloaded from WWW*

Output: *Searchable query web forms*

1. The query will be fired and web pages will be downloaded.

2. The non-searchable web pages will be extracted and discarded.

3. The downloaded searchable web forms will be matched with the sample web forms by the inference mechanism.

3.1 The set of rules and facts which can be applied on the given web form will be decided.

3.2. The shortlisted set of rules and facts will then get stored in the working memory.

3.3. The current web form will be picked and one by one the rules from the working memory will be applied.

3.4. When by the execution of any rule the goal state is reached, then that web form will get selected.

3.5. The web forms will be grouped as SSBWF and MSBWF.

3.6. The details of web forms such as URL, keywords will be extracted.

Figure 5. 9. Web Form Extraction and Inference Mechanism

5.2.3. Semantic Graph Creator and Mapper

The extracted and processed web forms from previous module is now fed as input to this module. The output of the module is the mapped semantic graphs of web forms. This module maps the extracted web forms with the sample web forms. The URL and the keywords found in the selected web forms are linked to the web forms in the graph. The frequency of the keywords found in the web forms also get stored as a tuple (keyword, count) in the web form semantic graph as shown in Fig 5.10. The web forms are filled with a default value "default_X" and are auto submitted. The URL's generated are then captured and get stored as the URL templates in the web form semantic graph. When end user issues the query these default values are replaced by the end user's query keywords and the fresh results are be fetched from the website's server dynamically.

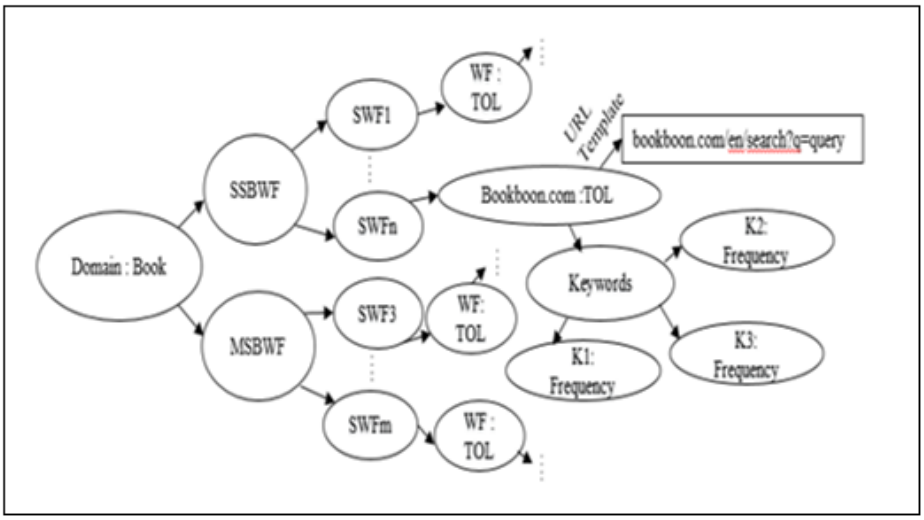

Figure 5. 10. Semantic Web Form Graph

5.2.4. Ranking of Web Forms

The semantic web form graphs generated above are provided as input to this module and the ranked web forms are generated as output. Three components have been used for ranking the web forms. The first component is the similarity between the downloaded web forms and the sample web forms stored in the knowledge base. If the percentage of similarity of web form and the sample web form is high, then that web form's contribution in ranking is also high. The downloaded web pages containing the web forms also contain the URL's of other web pages. These web pages are also crawled and if they hold the query interfaces or web forms of the same domain then these pages get downloaded and are linked to their parent URL web form in the semantic web form graph as shown in Fig 5.10. The number of outlinks in a URL also get stored in the URL node as a tuple (URL, total number of outlinks (TOL)) and it also contributes to ranking of web forms. For example, bookboon.com has a link to subsites.bookboon.com. More outlinks result in raising the rank. The third component in ranking the web form is the distribution of the benefit count of the inlink or the parent node in the web graph among its child nodes.

The steps for ranking the web form are:

1) **Keyword Weight Calculation and Similarity Finder**

The keywords and the frequency of the keywords found in the extracted web forms are be compared with the sample web forms. The total weight of the keywords are calculated by adding the frequency of all keywords found in the web form. If the

frequency of the i^{th} keyword of web form under processing is more than the frequency of the i^{th} keyword of sample web form, then the frequency of the i^{th} keyword of the sample form is taken else frequency of i^{th} keyword of form under processing is taken. The weight is calculated using the proposed formulae given in equation 1.

$$\text{Weight}^{u} = \sum_{i=0}^{count} if(\text{Kw\_freq}_{\text{iu}} > \text{Kw\_freq}_{\text{is}})\ then\ \text{Kw\_freq}_{\text{is}}\ else\ \text{Kw\_freq}_{\text{iu}} \quad (1)$$

Weight^{u} : It is the total weight of keywords for the web form "u".

Count : It is the total number of keywords in the sample form "s".

Kw_freq_{iu} : It is the frequency of i^{th} keyword in the web form "u".

Kw_freq_{is} : It is the frequency of the i^{th} keyword in the sample web form "s".

So, if the frequency of a keyword in the web form is more than the frequency of that keyword in the sample form then the frequency of keyword in the sample form "s" is taken. The similarity between the web form and the sample web form is calculated using the proposed formulae given below in equation 2.

$$\text{Sim}^{u} = \frac{\text{count\_kw}_{\text{u}}}{\text{count\_kw}_{\text{s}}} * \frac{\text{weight}_{\text{u}}}{\text{weight}_{\text{s}}} \quad (2)$$

Sim^{u} : It is the similarity of the web form "u" with one of the sample form "s".

count_kw_{u} : It is the total number of keywords found in web form "u".

count_kw_{s} : It is the total number of keywords found in sample web form "s".

Weight_{s} : It is the weight of all the keywords found in sample web form "s".

Weight_{u} : It is the weight of all keywords found of web form "u".

The value of the similarity will vary from 0 to 1. The similarity of the forms is stored in the edges of the semantic web form graph.

2) Benefit Calculation using the Outlinks

It calculates how beneficial a link or URL can be. If a link further leads to similar web forms, then that link is very beneficial as shown in Fig.5.11. The root node "Book" represents the chosen domain "book". The children of that node are SSBWF (Single Search Box Web Form) and MSBWF (Multiple Search Box Web Form). The child "https://www.bookadda.com/ "of SSBWF has one outlink "https://www.acadzone.com/" but "https://www.goodreads.com" has no outlink. So https://www.bookadda.com/ is more beneficial than but "https://www.goodreads.com". This information about outlinks is stored in the semantic web form graph along with the other details of web forms. The benefit of the web forms is calculated by the help of the proposed formulae given below in equation 3.

$$\text{Benefit}^{u} = (1-d) + \frac{\text{d}((\text{Sim}_{\text{OL1}}+\text{Sim}_{\text{OL2}}+\cdots\text{Sim}_{\text{OLn}})*\text{n})}{\text{Length}} \quad (3)$$

Benefit^{u} : It is the benefit count of web from "u".

N : It is the total similar outlinks of "u" which are stored in the semantic web graph.

Sim_{OLn} : It is the similarity of the n^{th} outlinks of web form "u".

Length: It is the total number of external outlinks found in web page

d : It is the damping factor which controls the value of Benefit_{u}.

The dampening factor is used to make sure that the benefit accidentally doesn't end up with infinite values. The value of d will be taken as 0.85.

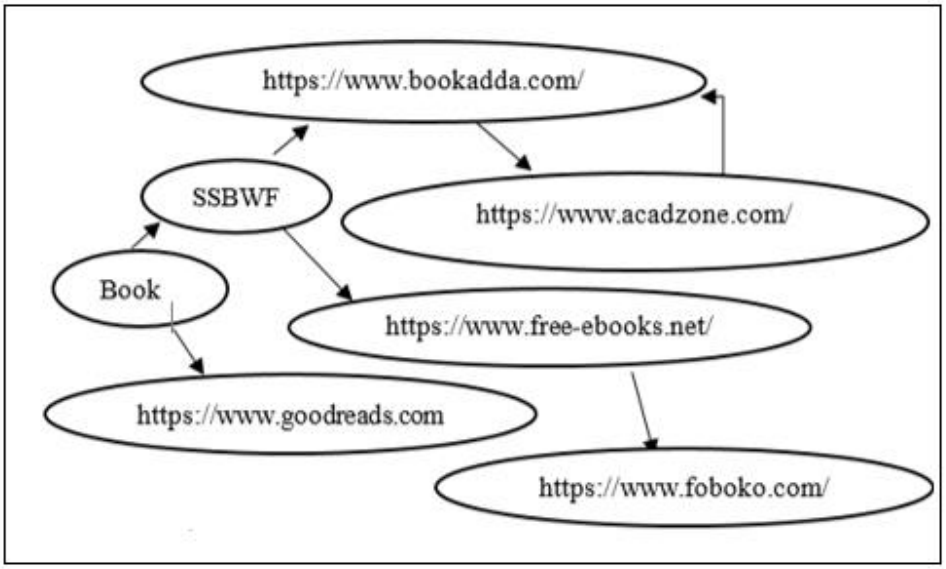

Figure 5. 11. Inlinks and Outlinks of Web Forms in Web Form Semantic Graph

3) Rank Calculation

The rank of the web form is calculated by using degree of similarity between web form and the sample web form, benefit of web page and the share of benefit from the parent of web form. The rank is calculated by the proposed formulae given in equation 4.

$$\mathrm{PR(u)} = (1 - d) + \mathrm{d(Benefit(u)} + Sim(u) + BC_{parentshare}) \qquad (4)$$

PR (u): It is the final page rank of the web form "u".

Benefit (u): It is the benefit count of the web form "u".

Sim (u): It is the similarity of the web form "u".

d : is the damping factor having value 0.85.

The benefit of the parent nodes of web form "u" will be shared among their child nodes. So, $BC_{parentshare}$ is the share of benefit count of web form "u" from its parent nodes. It is calculated by the formulae given in equation 5. Here

Benefit (P1) : It is the benefit count of first parent of web form "u".

Child_Total (P1): It represents the total children of first parent of "u".

$$\mathrm{BC\_parent\_share} = (\ \mathrm{Benefit(P1)/Child\_Total(P1)} + \mathrm{Benefit(P2)/Child\_Total(P2)} \ldots\ldots + \mathrm{Benefit(Pn)/Child\_Total(Pn)}\ ). \qquad (5)$$

5.3. Expert System for Data Region Extraction and Integration (ESDREI)

The input to Expert System for Data Region Extraction and Integration (ESDREI) is the downloaded structured web documents and the generic URL templates generated by WFER. The generic URL templates get filled with the user query keywords and the results pages are downloaded. These result web pages are the structured web documents which contains the data from the web server databases.

The architecture of Expert System for Data Region Extraction and Integration (ESDREI) is shown in Fig. 5.12. This module uses the fuzzy rule based expert system which is used for defining the domain and extracting the data region. Using the rules data region is extracted and the details get stored in a graph called semantic linked graph. The steps of the proposed technique are given below.

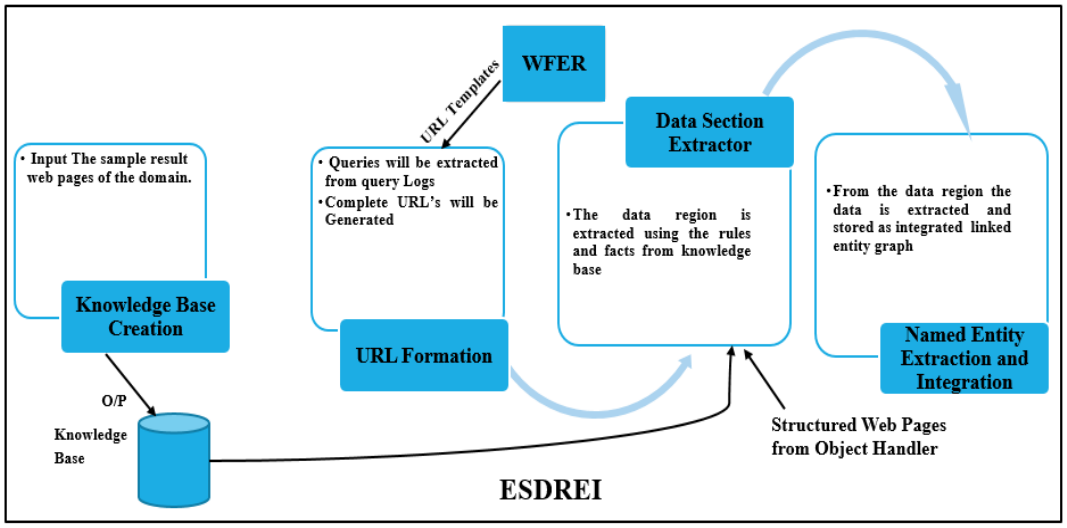

Figure 5. 12 Proposed Architecture of ESDREI

The proposed technique is followed by following four steps.

- Knowledge base creation

The information about the properties of an object/domain is stored. This information is converted into the rules, which will help in extracting the data region from the result web pages.

- URL formation

The specific URL's are created by completing the generic URL templates. The query logs are processed to generate the complete URL's. These web pages contain the data from the web server databases.

- Data section extractor

Using the knowledge base, the data section is extracted from the result pages. The data sections contains the data from the hidden web.

- Named entity extraction and integrator

The data from the data sections extracted above is pulled out and get stored as semantic linked graph.

5.3.1. Knowledge Base Creation

The knowledge base is created to identify the data region automatically from the web pages. The input to this module is the domain knowledge and the high-quality information about the possible data containing sections in the web pages and hence knowledge base is created as shown in Fig 5.13.

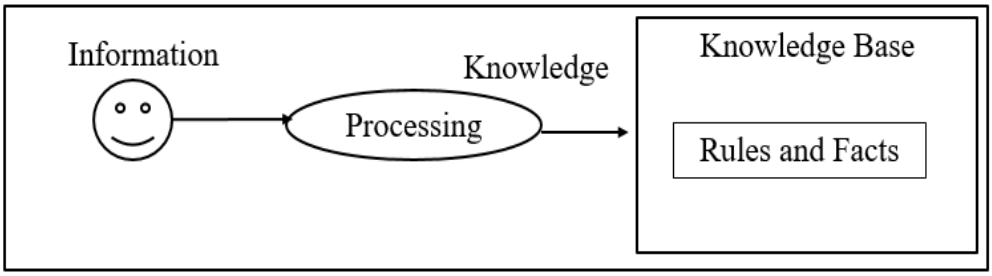

Figure 5. 13. Knowledge Base Creation

The knowledge is represented by the fuzzy rule-based system. The knowledge base contains the data, facts and the rules about the domains. The domain knowledge contains the attributes defining the object. For example, the book has attributes name, title, author, subject, publisher, volume, year of publication, price. So the description about any named entity is collected and the facts are created. For example the facts about the book entity are given in Table 5.4.

Table 5. 4. Facts defining the domain

Fact ID	Facts
F1	Written_by (book, author)
F2	Has (book, name)
F3	Has (book, title)
F4	Has (book, subject)
F5	Published (book, year)
F6	Has (book, cost)
F7	Has (book, volume)
F8	Has_unique (book, ISBN)

The high-quality information about the sections of a web pages that display the hidden web data is provided to the system. The data collected is converted into knowledge or rules. Later these rules are used by the rule interpreter. As per the information provided to the system few result web pages display the data from hidden web databases in a tabular form. From these result tables, few tables have the header which clearly indicates that it is the hidden data containing section as shown in Fig. 5.14.

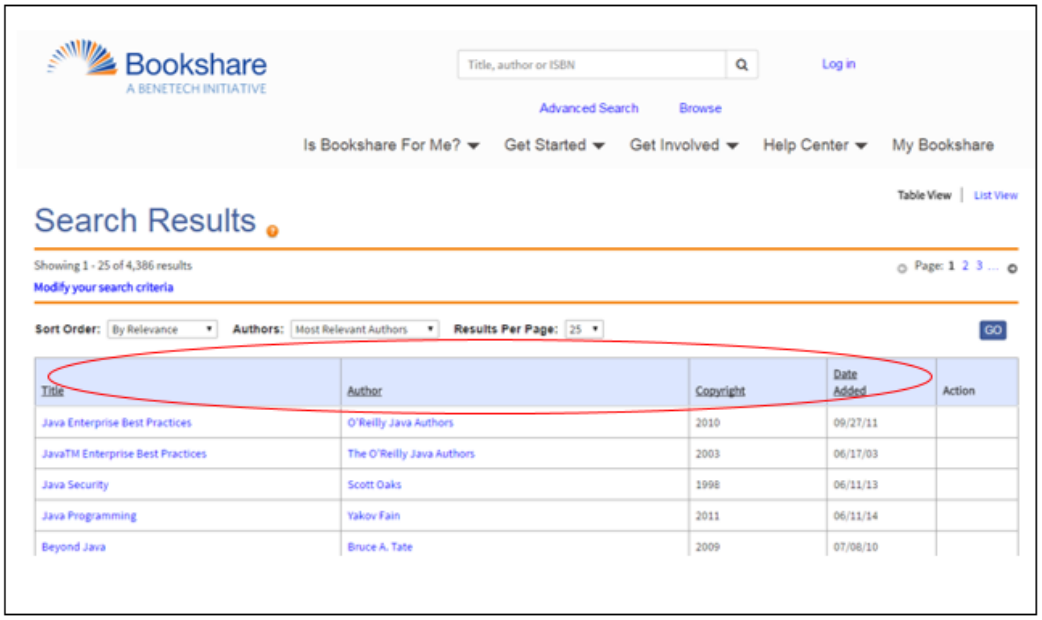

Figure 5. 14. Result Web page containing the <th> tag in <table> tag

The rules R1 to R7 are constructed which will help in automatic extraction of such data sections from the result web pages as shown in Table 5.5. The header of such tables

should contain the attributes describing the entity. According to these rules the table header should have some of the attributes describing the book entity.

Table 5. 5. Rule for Data Extraction

Rule ID	Rule
R1	Has (th, title) ^ Has (th, author)→Book (table)
R2	Has (th, title) ^Has (th, subject) →Book (table)
R3	Has (th, title) ^Has (th, year) →Book (table)
R4	Has (th, author) ^ Has (th, subject) →Book (table)
R5	Has (th, author) ^Has (th, year) →Book (table)
R6	Has (th, subject) ^ Has (th, year) →Book (table)
R7	Has (th, subject) ^Has (th, year) ^Has (author) ^has (title) →Book (table)
R8	∃ table (∀ tr (Has (tr, author) v Has (tr, by)) ^ (Has (tr, price) ^ Has (tr, followed (price, : number))) v Has (tr, ISBN) →Book (table))
R9	∃ul or ol (∀ li (Has (li, author) v Has (li, by)) ^ (Has (li, price) ^ Has (li, followed (price: number))) v Has (li, ISBN) →Book (li))
R10	∃div ((Has (div, author) v Has (div, by)) ^ (Has (div, price) ^ Has (div, followed (price: number))) v Has (div, ISBN) →Book (div))

As per the information provided to system, certain times the data is displayed in the table but the header is not there in the table tag as shown in Fig. 5.15.

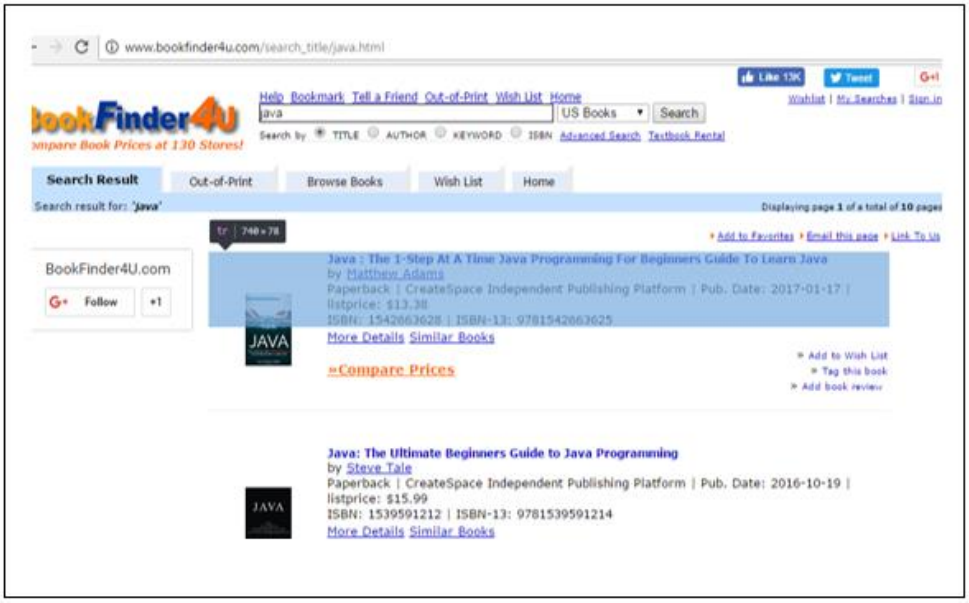

Figure 5. 15. Result Web page containing no <th> tag in <table> tag

All the row of the table should contain the attributes describing the object. So there should exist some table, whose all rows has the attributes describing the entity book. So, in order to identify such sections the rule R8 is constructed as shown in Table 5.5. As per the information provided by to the system data from the databases can be displayed in the list as shown in Fig. 5.16.

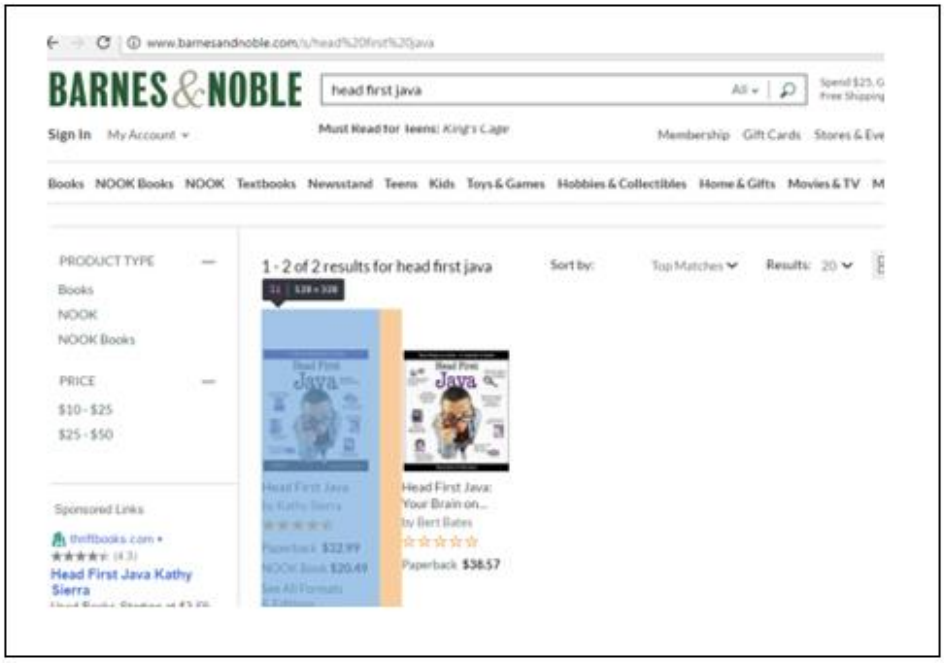

Figure 5. 16. Result Web page containing the data in <li> tags

There should exist some unordered or ordered list, whose list item should have the listing of the attributes describing the entity book. For the extraction of such section of a web page the rule R9 is given in Table 5.5.

The information provided to system related to data regions, the data from the databases can be displayed in the diversion (div) tag as shown in Fig. 5.17. All the tags of the html page are in a hierarchy. The tags are taken in descending order of their depth. For the extraction of such section of a web page the rule R10 is given in Table 5.5.

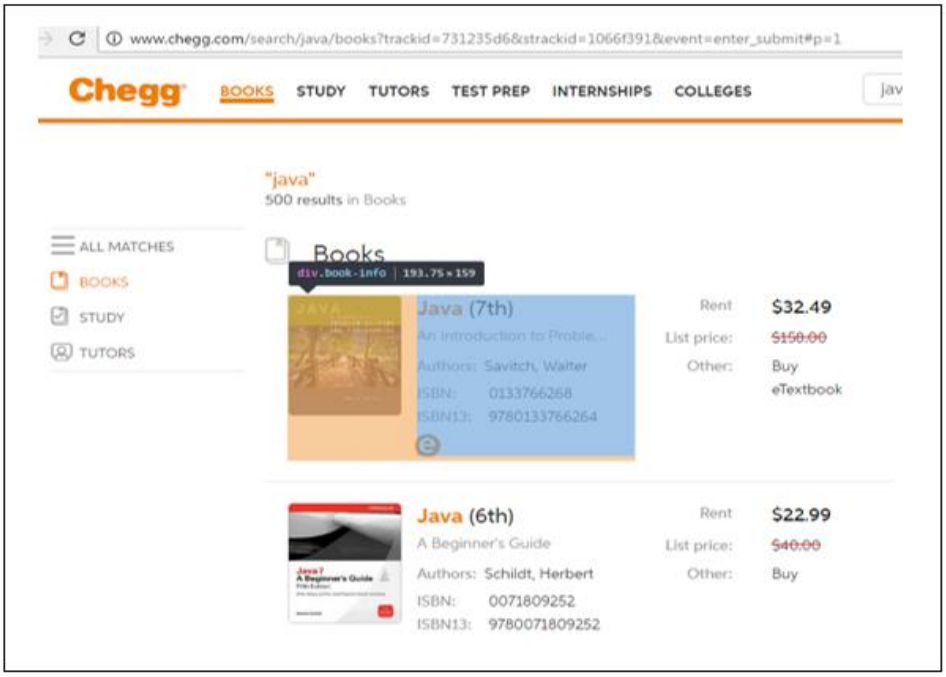

Figure 5. 17 Result Web page containing the data in tags

5.3.2. URL Formation

The generic URL templates are now converted into complete URL's of rest pages. The input to the URL formation is the generic URL templates and the values of the attributes fetched from the query logs. The URL templates are the generic URL's and contains the default value "default_x" as shown in Table 5.6.

The default value is now replaced by the specific values of the keywords. The frequently used keyword values for the domains are collected from the query logs. The query logs contain many keywords values which are issued by the users. The query logs are processed and the frequently used keywords are now extracted. The default value "default_x" is replaced by the keywords extracted and the URL's are created as shown in Table 5.7. These web pages will act as an input to the Data Section Extractor.

Table 5. 6. URL templates for bookfinder4u.com

URL ID	URL Template
1	http://www.bookfinder4u.com/search_author/default_x.html
1	http://www.bookfinder4u.com/search_title/ default_x.html
1	http://www.bookfinder4u.com/search/ default_x.html
1	http://www.bookfinder4u.com/compare.aspx?isbn= default_x
1	http://www.bookfinder4u.com/uk/title/ default_x.html
1	http://www.bookfinder4u.com/uk/author/ default_x.html
1	http://www.bookfinder4u.com/uk/keyword/ default_x.html
1	http://www.bookfinder4u.com/fr/title/ default_x.html
1	http://www.bookfinder4u.com/fr/author/ default_x.html
1	http://www.bookfinder4u.com/fr/keyword/ default_x.html
1	http://www.bookfinder4u.com/ca/title/ default_x.html
1	http://www.bookfinder4u.com/ca/author/ default_x.html
1	http://www.bookfinder4u.com/ca/keyword/ default_x.html
1	http://www.bookfinder4u.com/de/title/ default_x.html
1	http://www.bookfinder4u.com/de/author/ default_x.html
1	http://www.bookfinder4u.com/de/keyword/ default_x.html
1	http://www.bookfinder4u.com/jp/title/ default_x.html
1	http://www.bookfinder4u.com/jp/author/ default_x.html
1	http://www.bookfinder4u.com/jp/keyword/ default_x.html

Table 5. 7. URL's generated

URL ID	URL
1	http://www.bookfinder4u.com/search_author/horstman.html
1	http://www.bookfinder4u.com/search_title/java.html
1	http://www.bookfinder4u.com/search/programming.html

5.3.3. Data Section Extractor

The input to the data section extractor module is the result web pages and the knowledge base which contains the rules as shown in Fig. 5.18.

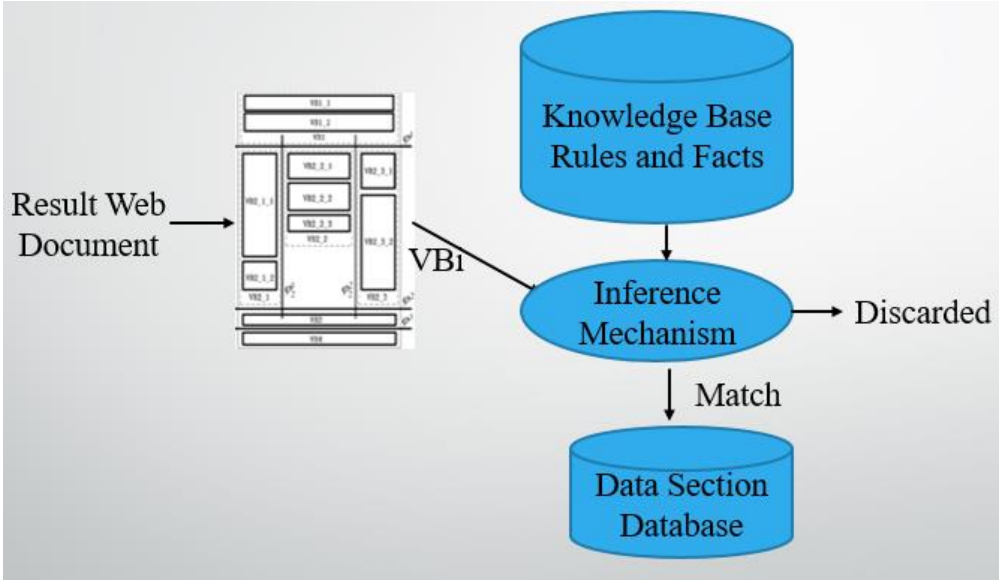

Figure 5. 18. Data Section Extractor

The current result web page gets stored in the working memory. The data section extractor or the inference mechanism (IM) reads the details or tags of the current web page and it then read the rules and facts from the knowledge base. Now one by one the rules are compared with the current web page's expected data containing sections. If the section matches with the any of the rules then that section is pulled out and is stored in data section database. The data sections is now stored in the memory for the extraction of records from the hidden web.

5.3.4. Named Entity Extractor and Integrator

The input to this module are the facts and the extracted data regions. There are two issues with the data lying on the web server databases. The first issue is that the data extracted about a single object from different websites might give details of different properties of the object. For example, some websites might provide the title, author, price and ISBN of a book. Some websites might give title, author, price, ISBN, volume and year of publication etc. of a book. The web servers stores the data using the relational databases, which have a fixed schema. The second issue is that the details of a single entity or object can be present in more than one web server's database. For example the details of "Complete Reference" book of subject java is present on multiple

web servers. So it leads to redundancy of the data on the web. This proposed technique resolves two issues by providing flexibility of storing different properties and removing redundancy of data. A node of a URL, which displays the data from hidden web is created and it is connected to its all records. The records are stored as separate nodes. All the properties of a record are linked to the record node. The properties are displayed in rectangles. The semantic association of the property of record or object is described using the relationship. The relationship is described using the arcs. As no pre described fixed schema or structure of the records is required in the proposed technique, the first issue addressed above is resolved. If there are two URL's which provide details about two similar books, then the nodes of those two records are created but information is attached to only one node. The second record node is connected to first record node using the similar relationship. If there is some information which is present only in second record from URL2 then only the unique property detail get attached to it. So the second issue of information redundancy is also resolved. All the details of the entities in the WWW are stored once and the relationship is stored if two entities are linked to each other as shown in Fig 5.19. For example "Complete Reference" book of java is written by "Herbert Schildt" which itself belongs to class "person". So all the information of "Herbert Schildt" will be stored only once and the books written by him will be directly linked with the relationship.

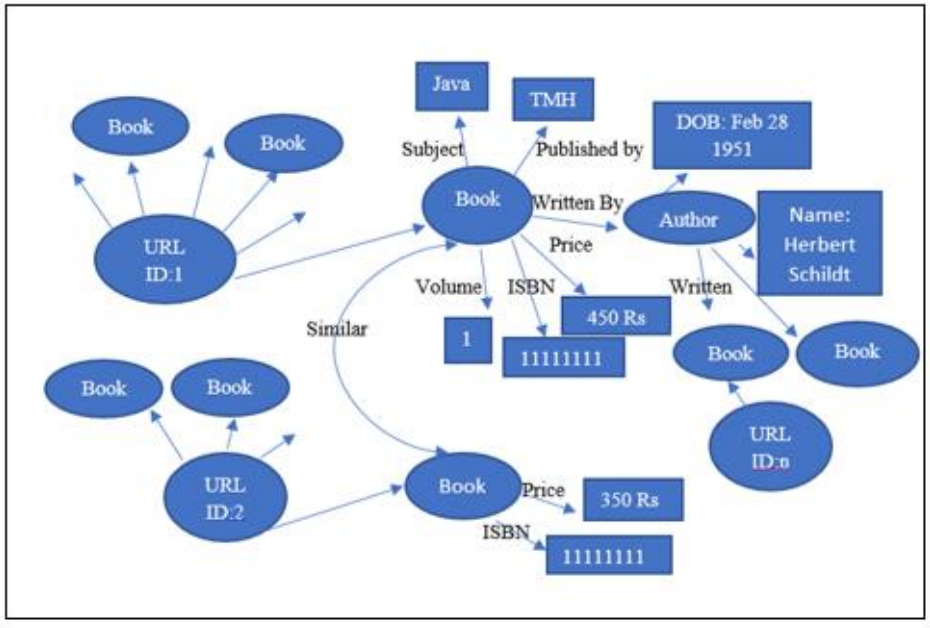

Figure 5. 19. Semantic Linked Graph of Hidden Web Data

5.4. CONCLUSION

This chapter presented the details of the last two steps used in the proposed system. These steps help in extracting the web forms, which acts as a doorway to hidden web, ranking of web forms and integrating the hidden data from multiple web sources. This proposed work has been carried out in two modules WFER and ESDREI. WEFR processes the web pages containing the query interfaces and rank them using the fuzzy rule based system. WFER has been carried out in four steps namely, Knowledge Base Creation, Web Form Extraction and Inference Mechanism, Semantic Graph Creator and Mapper and Ranking of Web Forms. The output of WFER are the ranked web forms with their details stored as a semantic graph. Semantic web form graph stores the keywords, the frequency of keywords and the generic URL templates.

ESDREI helps in identifying the data region from the result web pages and integrating the data from multiple web sources. The input to ESDREI are the generic URL templates generated by WFER and the structured web pages from the object handler. ESDREI is a fuzzy rule based system. It comprises of four steps namely, Knowledge Base Creation, URL Formation, Data Region Extractor and Named Entity Extractor and Integrator. ESDREI aggregates and removes the redundant data found in WWW databases. Next chapter presents the results thus obtained and also a detailed discussion concerning the results.

CHAPTER 6

RESULTS AND DISCUSSION

6.1. INTRODUCTION

The data from the hidden web is of very high quality and quantity. A major part of data from WWW is composed of Hidden Web but still very little data has been extracted from the Hidden Web. The critical observation of the literature revealed that the content of a domain is scattered across different types of web pages. These web pages can be part of surface web or the hidden web. Having the details scattered across multiple pages poses problem as the end user has to go through multiple web pages to extract the desired data. This decreases the user satisfaction level and increases the browsing time to extract the data. In order to address the concern thus raised "Dynamic Query Processing for Hidden Web Data Extraction (DQPHDE)" has been designed with the intention to extract, process and integrate the data spread across multiple web pages. The idea was to classify the different types of web pages then process them and help the end users. Different modules of DQPHDE were able to achieve this objective. The work was then tested and compared with the existing techniques and the results thus obtained are promising.

6.2. THE PROBLEM STATEMENT

With the evolution of internet and the growth of data, the first issue is that major part of data on WWW is stored in the database because it helps in managing and displaying the data on web pages easily. The second issue is that the data of a domain is scattered across multiple web pages. So duplication of data can be there. Because of these issues, the end users are not able to access the desired data easily and quickly.

As the existing hidden web aggregators are be able to fetch the data from the different databases or hidden web and store the processed data locally, there is a need for integrating and storing the processed data from the surface web as well as the hidden web at a single place in order to reduce effort and the browsing time of end users. For integrating and processing the data from surface web, the need of text mining, clustering and semantic web came into consideration. To extract data from different hidden web pages the need of an artificial intelligence approach is considered to be beneficial. The fuzzy rule based approach is included in the proposed architecture for fetching the result

sections from hidden web pages and these result sections are processed. The duplicate records are then removed and the information about an entity scattered across multiple web sources is integrated too.

The end users have to face many problems ranging from issuing many queries on both surface web and hidden web, fetching multiple web pages, processing many web pages and then only they are able to extract the desired data. A novel approach was thus proposed that helps the end users to overcome these problems. As mentioned already, Dynamic Query Processing for Hidden Web Data Extraction (DQPHDE) is divided into four phases namely Clustering using Graph and Relationship Extraction (CGRE), Semantic Based Text Mining and Summarization (STMS), Web Form Extraction and Ranking (WFER) and Expert System for Data region Extraction and Integration (ESDREI). The whole process clusters, summarizes, collects web forms, ranks web forms, and extracts hidden data from the different types of web documents of a domain as shown in Figure 6.1.

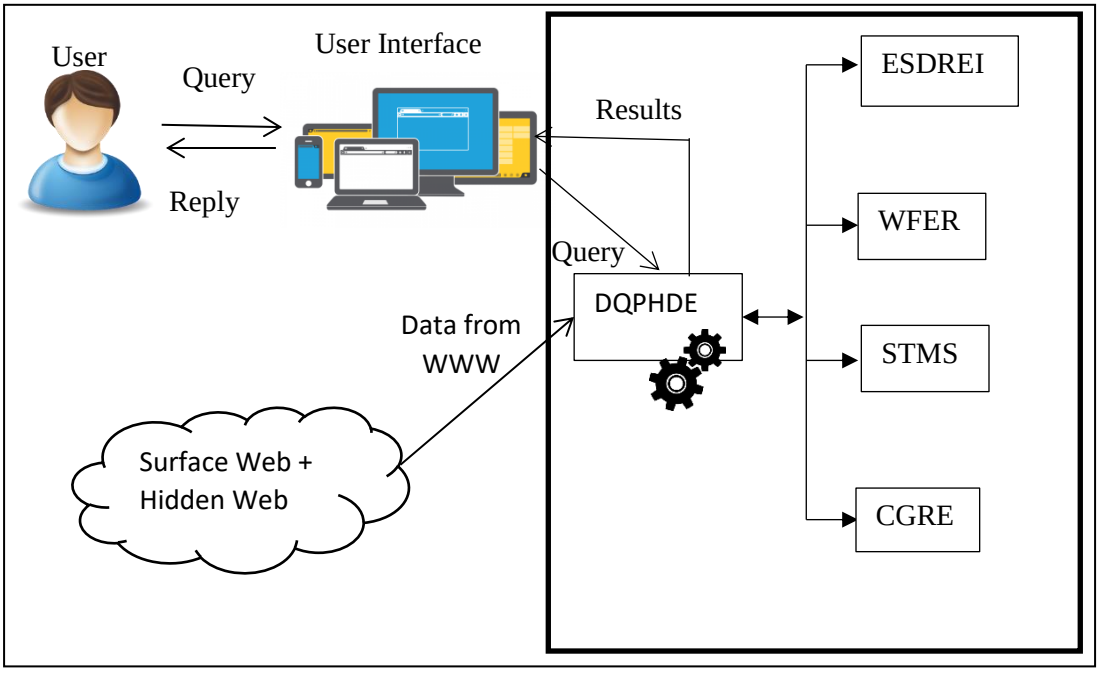

Figure 6. 1. The Proposed Architecture of DQPHDE

The upcoming section illustrates the implementation details of the proposed work.

6.3. IMPLEMENTATION DETAILS

DQPHDE is a system that uses clustering, semantic based text mining and fuzzy rule based system to extract the information about a domain from different types of web

pages and provides integrated and summarized information to the user. DQPHDE is implemented on Intel core i7 with 8GB RAM using Windows 10 using .NET C# as front end and SQL 2000 as backend.

6.3.1. Graphical User Interface

DQPHDE has successfully processed and integrated information about a domain from different areas of WWW i.e. hidden web and surface web. An Academic Search Engine prototype for the research scholars, has been created by the proposed technique. The test has been conducted on the research areas of the computer domain.

System Side:

The home page for the system administrator is shown in Fig 6.2. The home page has one input text box and one search button. When the query is issued and fired on home page then links of the web pages relevant to the query are returned.

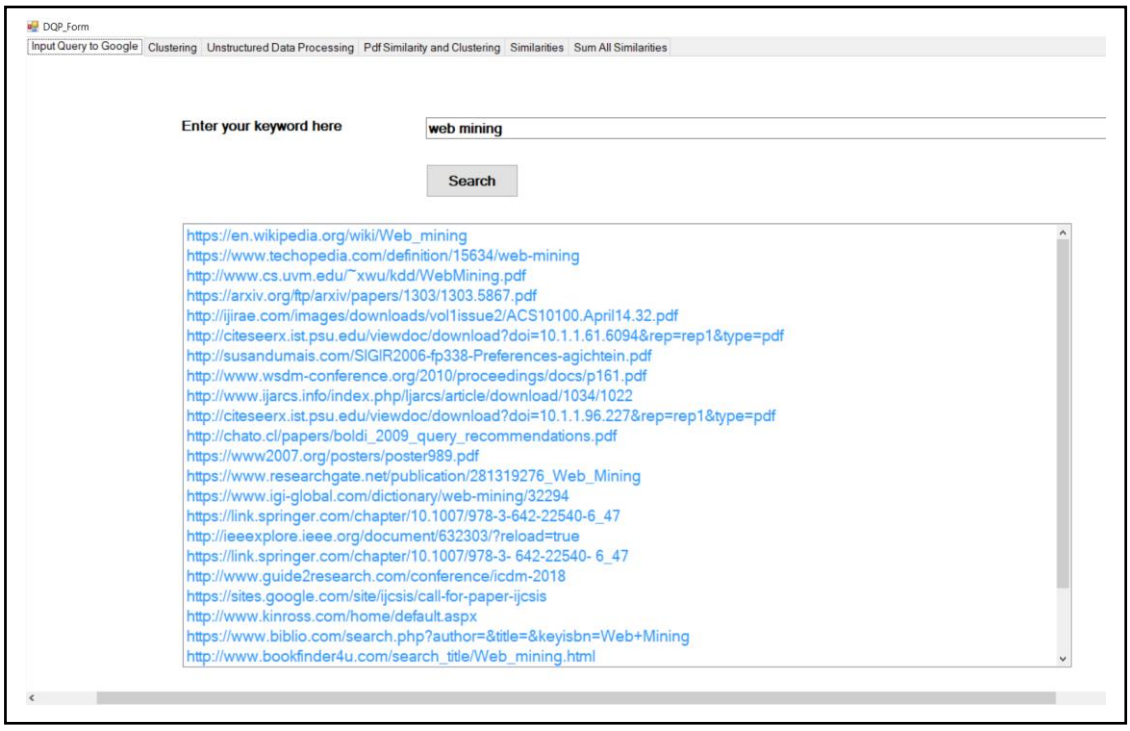

Figure 6. 2. Home Page for the System Administrator

Now the web pages get downloaded and are then processed by the different modules of the DQPHDE.

In the beginning, the downloaded web pages are classified as unstructured web pages, structured web pages, pdf or as query interface containing web pages as shown in Fig 6.3.

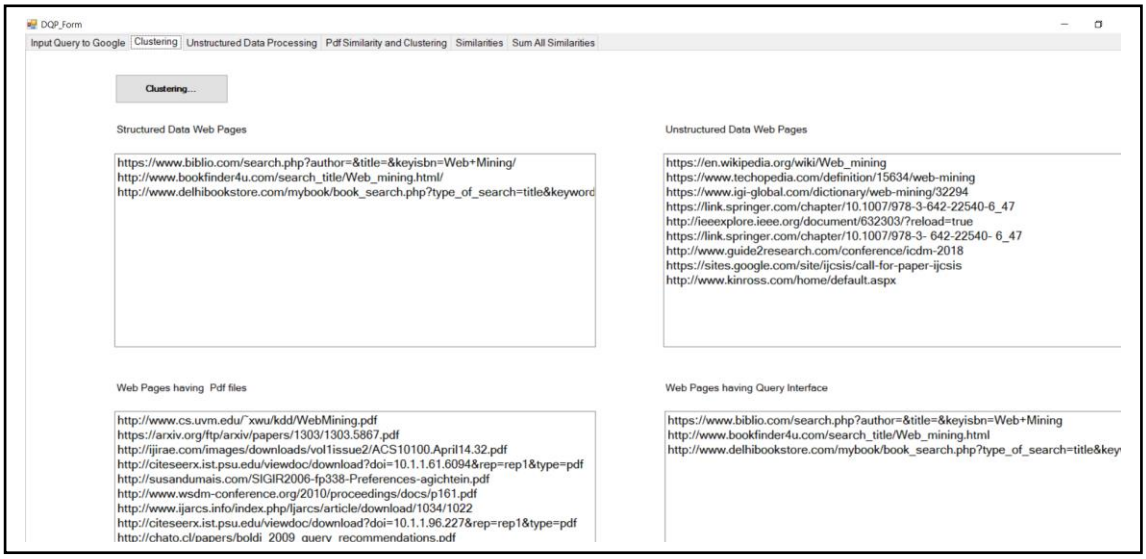

Figure 6. 3. Classification of Downloaded Web Documents

Next the web pages that contain the research articles are extracted and processed. The key features are extracted from the downloaded research articles. In the proposed work key features taken are area, author, journal name and year of publication. The downloaded research articles are then processed and the values of these key features are fetched as shown in Figure 6.4.

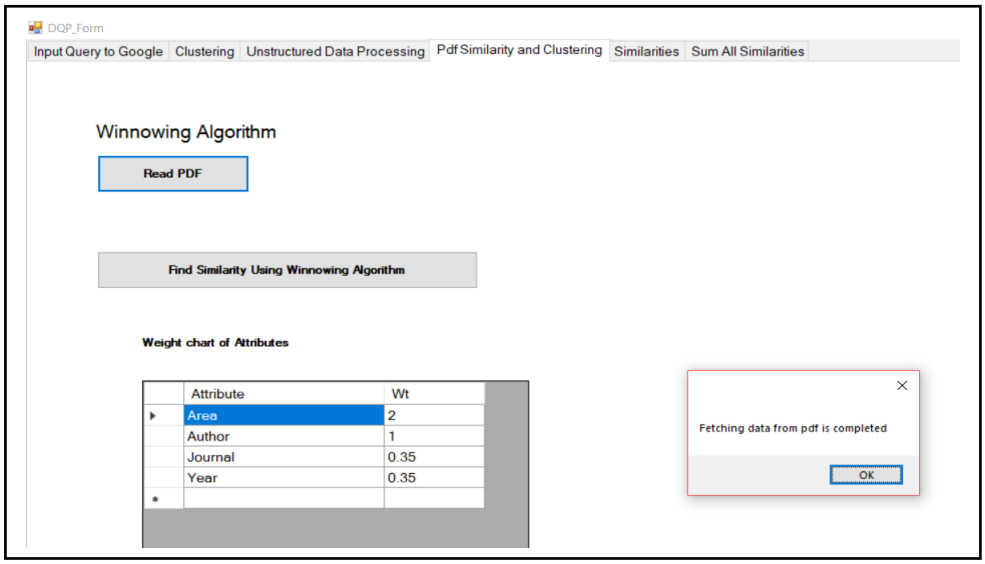

Figure 6. 4. Key Feature Extraction from Research Articles.

Then the similarity of the key features is calculated using the winnowing algorithm as shown in Fig 6.5.

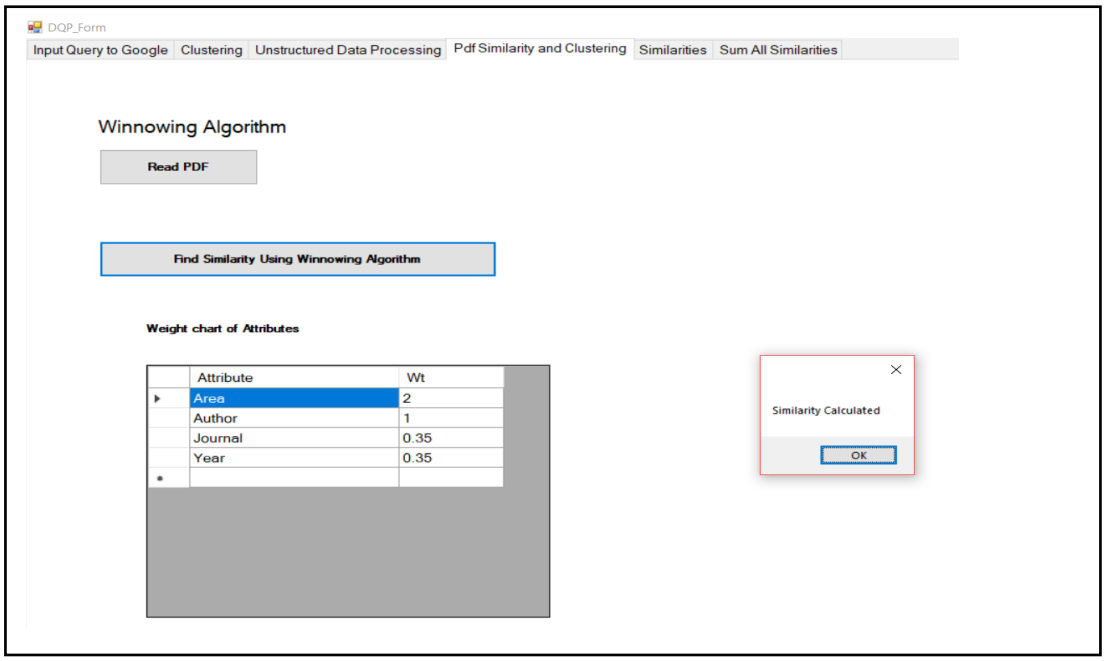

Figure 6. 5. Similarity Calculation of Extracted Key Features

The similarity of the key features using the Winnowing Algorithm is extracted and the results of similarity are shown Fig 6.6.

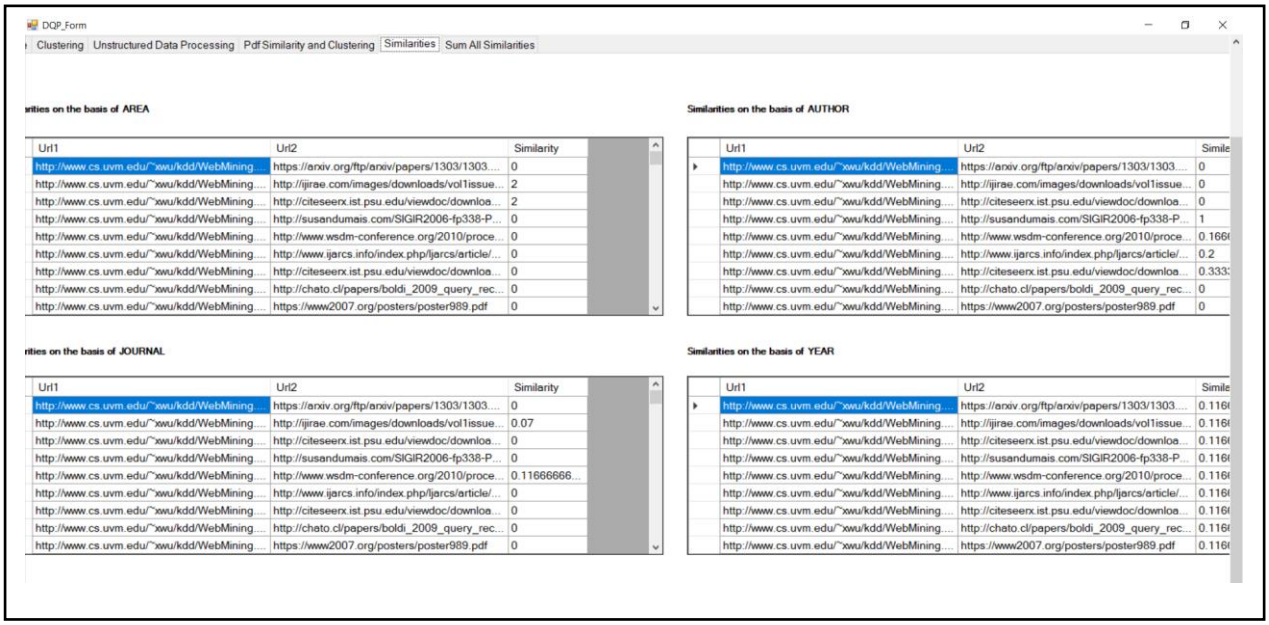

Figure 6. 6. Results of Similarity Calculation of Key Features

It is followed by the calculation of relationship strength between the research articles by the help of the proposed work as shown in Fig 6.7.

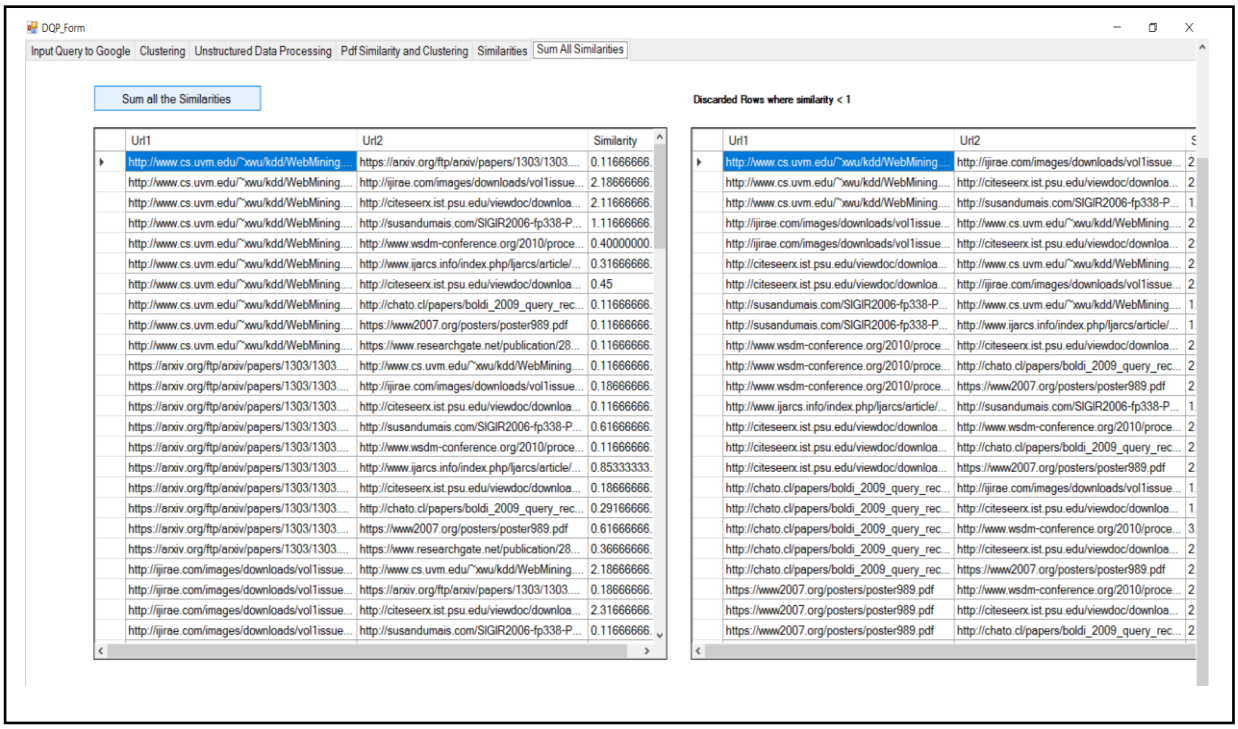

Figure 6. 7. Calculation of Relationship Strength between Research Articles

Now, on the basis of strength of relationship between the research articles the clustering of research articles is performed and the results of clustering are shown in Fig 6.8.

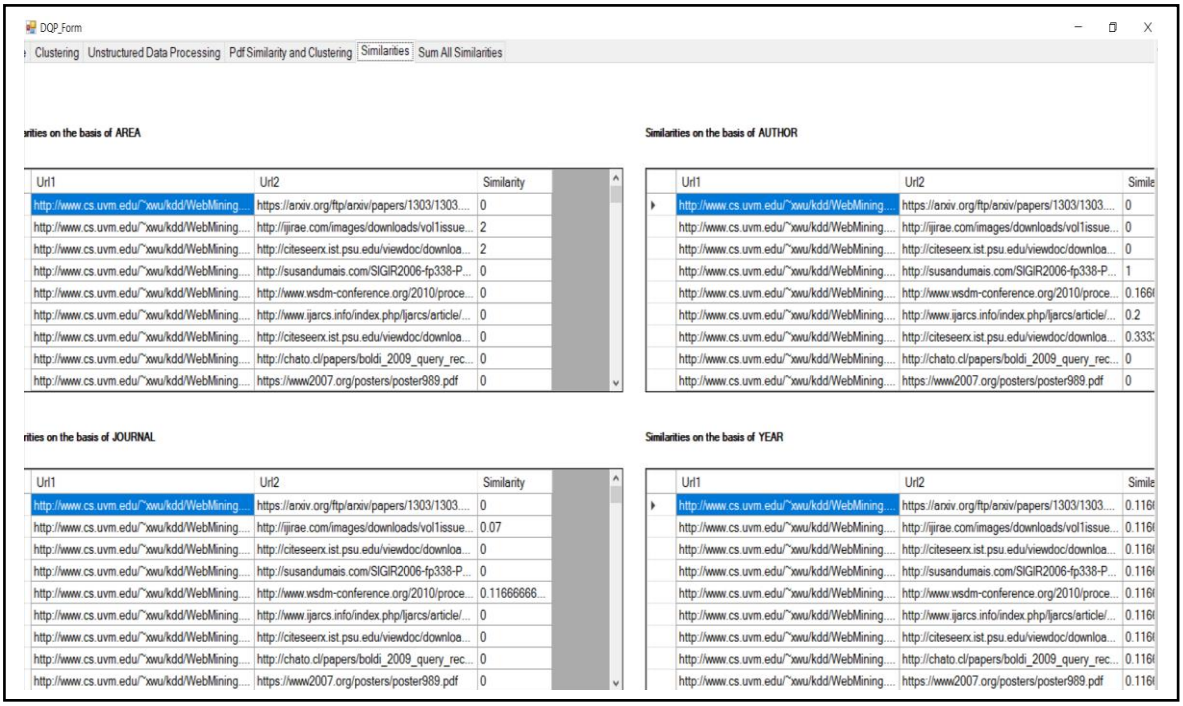

Figure 6. 8. Clustering of Research Articles based on Strength of Relationship

Now the unstructured web documents are processed. The unstructured documents are classified as call for papers, abstract only web pages, tutorials and irrelevant pages as shown in Fig 6.9. The irrelevant web pages are now removed from the document set.

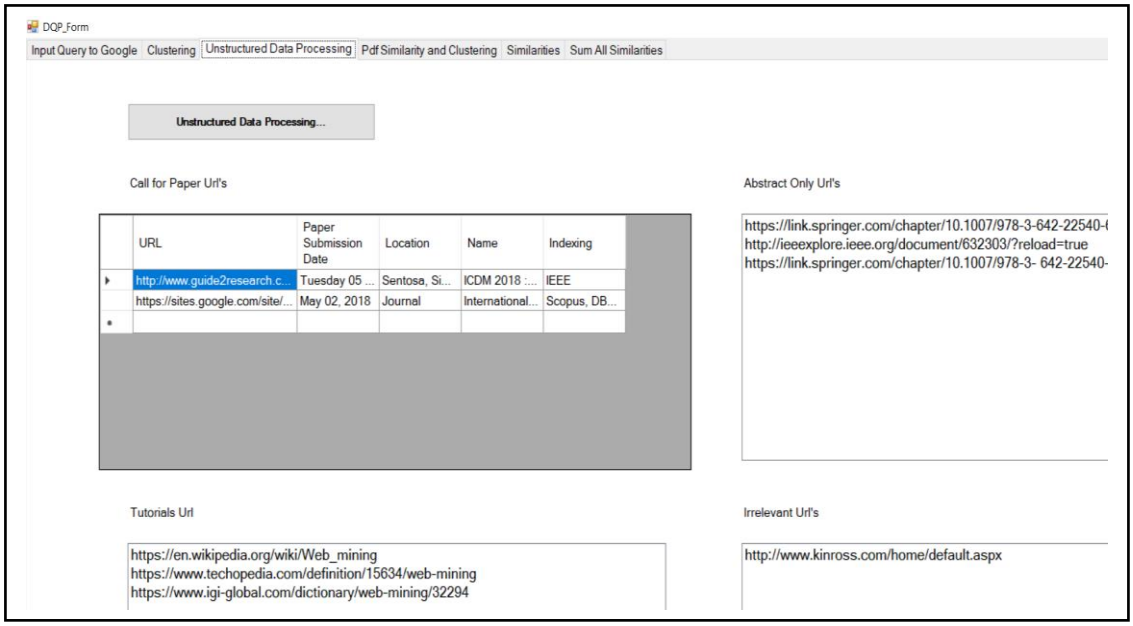

Figure 6. 9. Categorization of Unstructured Web Pages

The unstructured web pages that contains the tutorials are now processed by STMS module. The web documents are now get stored in the form of graph. The unique and common sentences are extracted now. The sentences from web documents are stored as nodes and relationship between those nodes as shown in Fig 6.10.

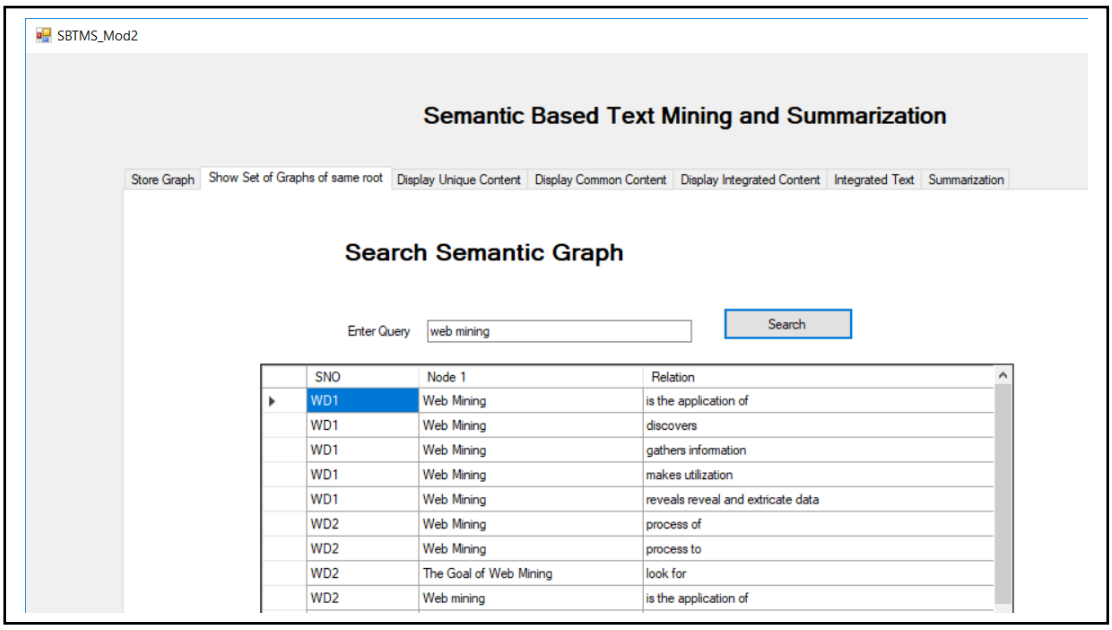

Figure 6. 10. Sample of Web Document Stored as Graph

It is followed by finding the unique content from the stored graphs of different web documents of a particular area as shown in Fig 6.11.

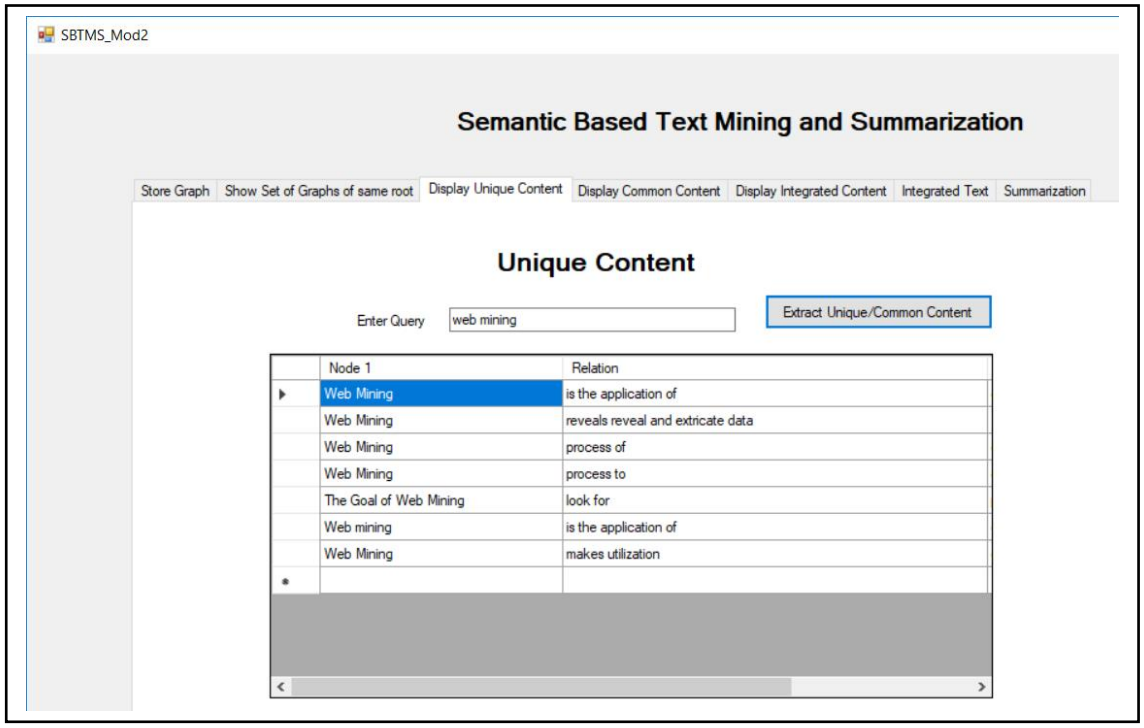

Figure 6. 11. Extracting Unique Sentences from the Graph of Word Documents

Next, the common content found from different web documents is extracted. The stored graphs of web documents are processed and the common content extracted is shown in Fig 6.12.

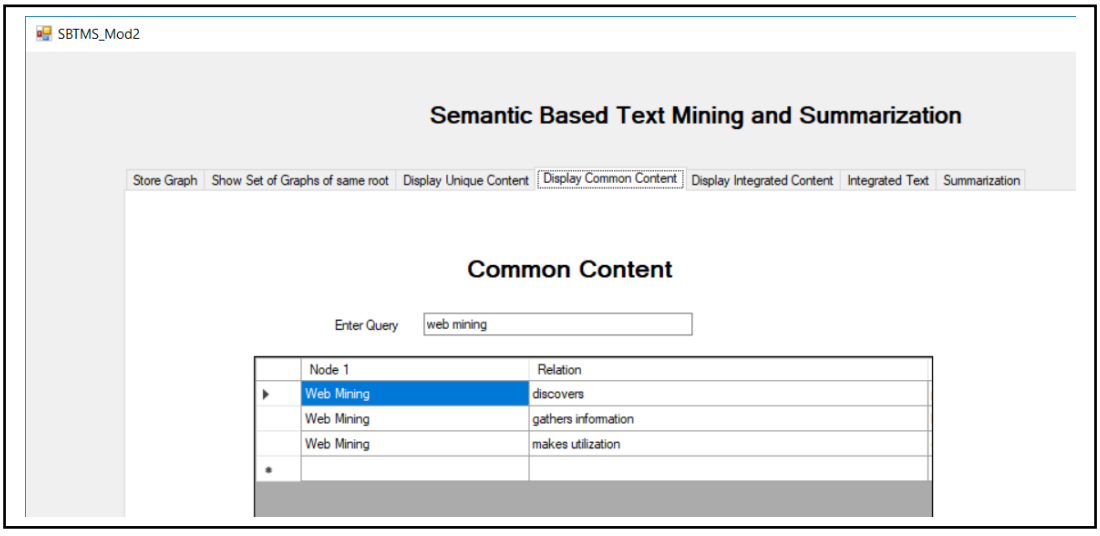

Figure 6. 12. Extracting Common Sentences from the Graph of Word Documents

It is followed by the integration of data from multiple web pages. The duplicate sentences are already removed when common content is extracted. The integration of data is done by merging the common and the unique sentences from the common and unique graph of web documents. The integrated content in the form of graph is shown in Fig 6.13.

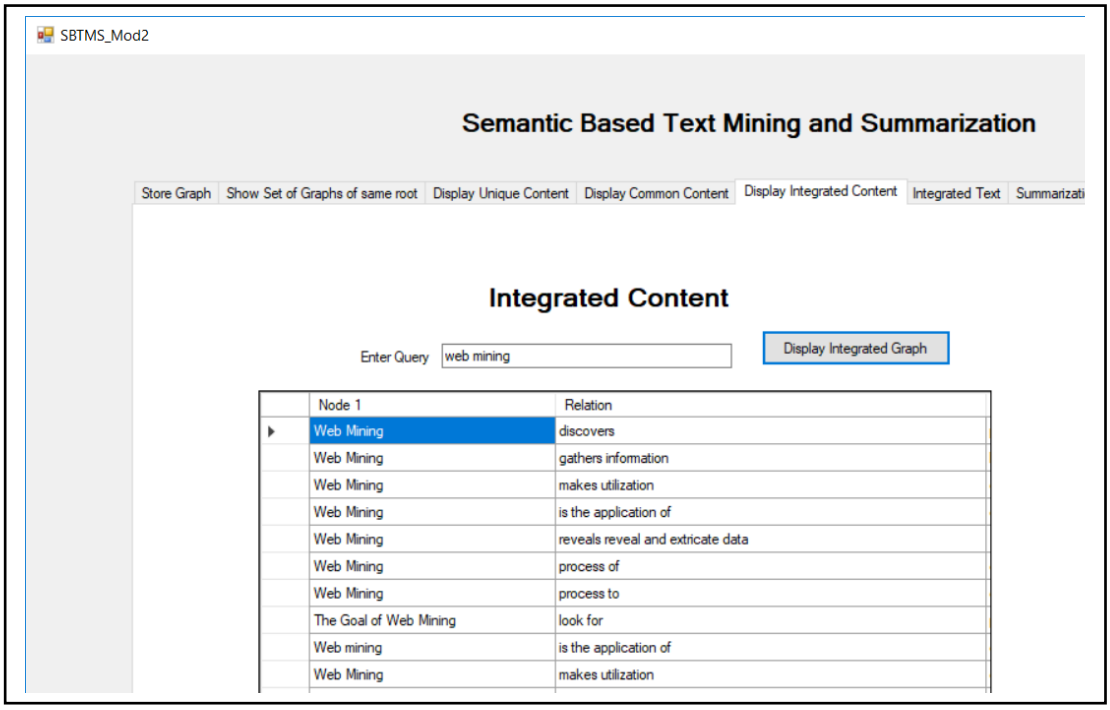

Figure 6. 13. Integrated Content from Web Documents in the Form of Graph

The integrated content in the form of graph is converted back to text for the end users by the help of semantic graph to text converter module of STMS. The integrated text is shown in Fig 6.14.

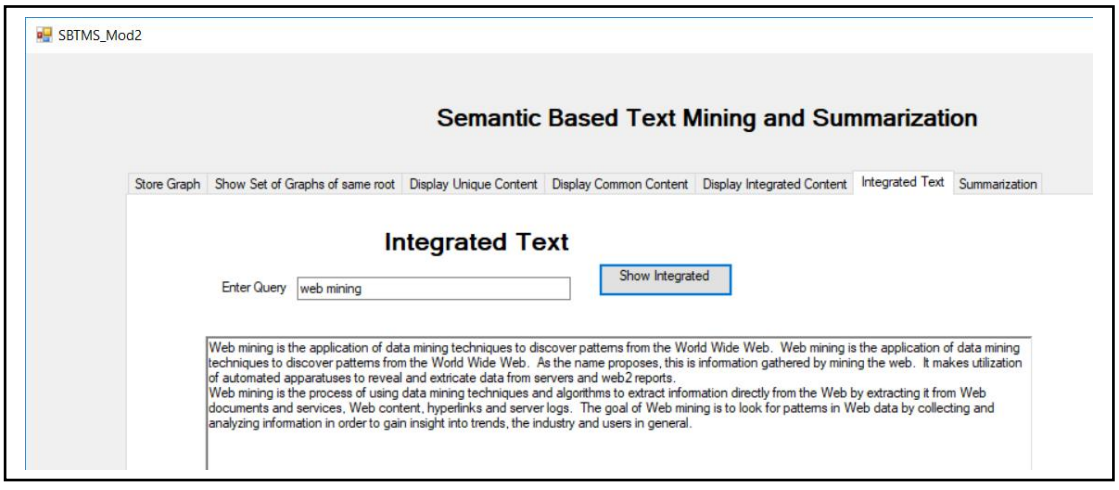

Figure 6. 14. Integration of Data from Multiple Web Pages

Integration is now followed by the creation of summary of the topic. The integrated graph contains the location of sentences as well as the frequency of the sentences. Using this information the summary of a topic is created as shown in Fig 6.15.

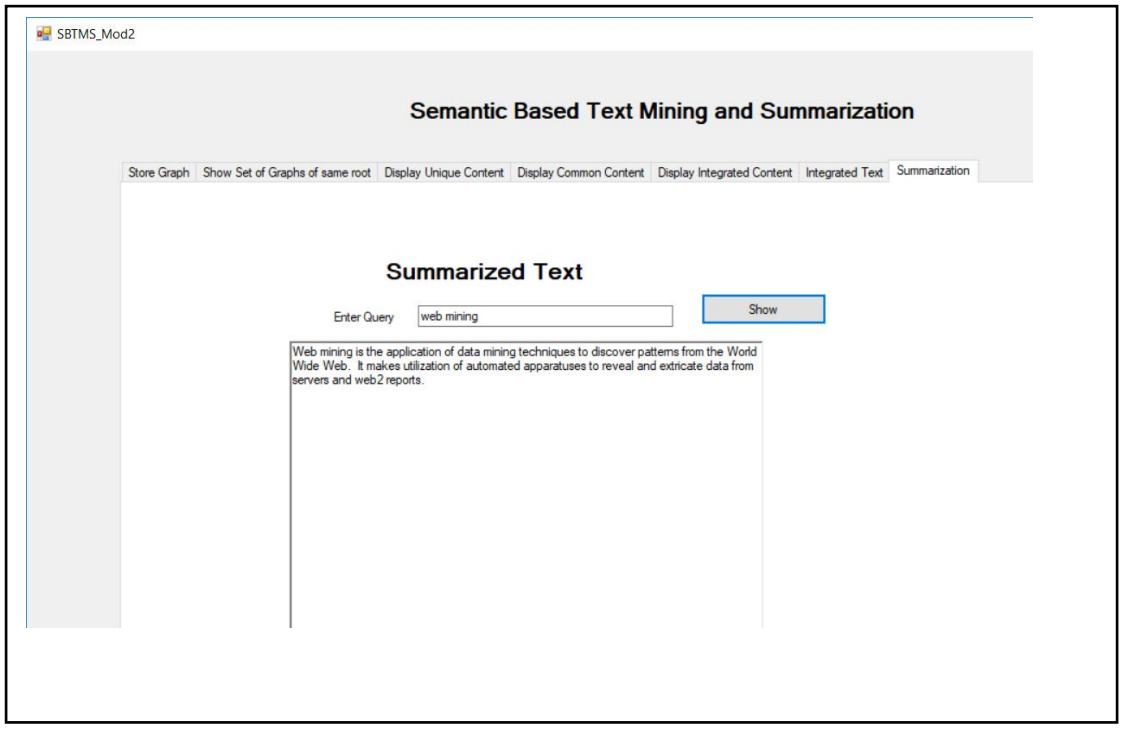

Figure 6. 15. Summary of Text from Multiple Web Pages.

In parallel, another module WFER is working. This module works on query interfaces. The first task of WFER is the extraction of query interfaces from the web pages. The web pages having the web forms act as input to WFER. These web forms are extracted as shown in Fig 6.16.

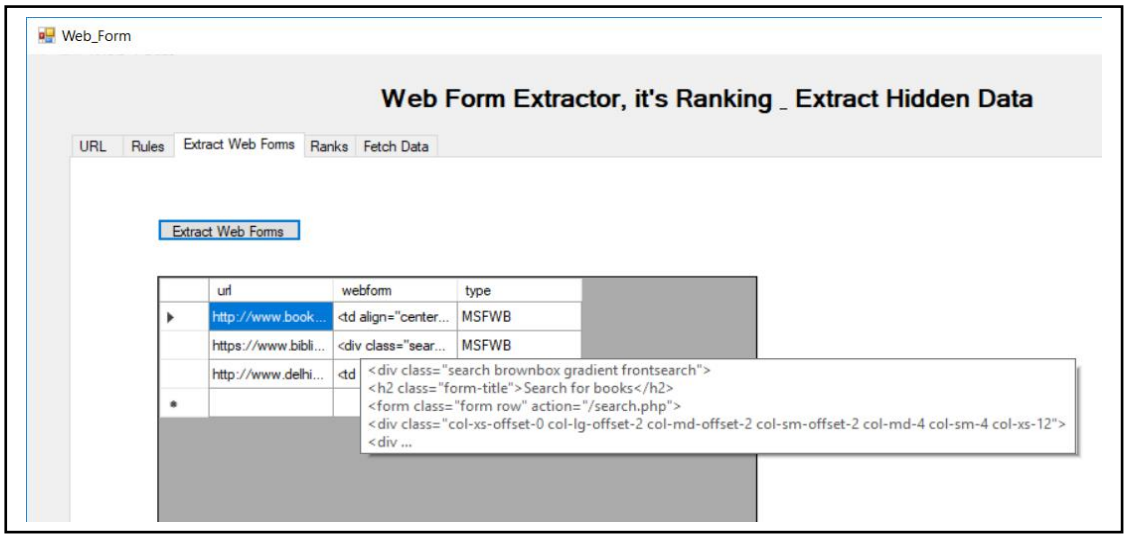

Figure 6. 16. Extraction of Web Forms from Web Pages

Next, the extracted web forms are ranked based on the inlinks, outlinks and the benefit of the web page which contains that web form as shown in Fig 6.17.

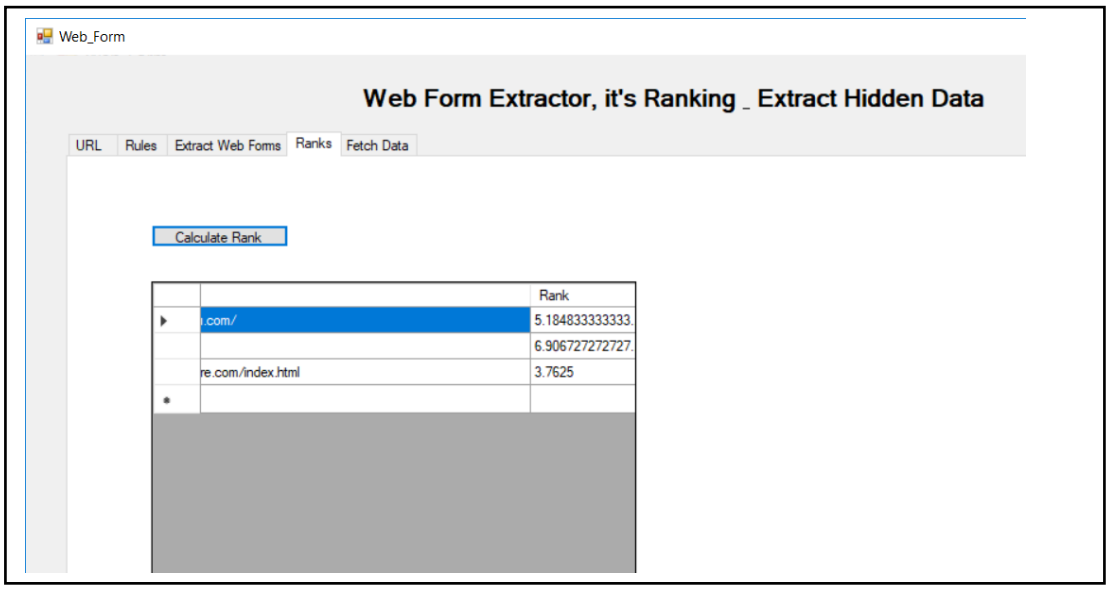

Figure 6. 17. Ranking of Web Forms

Now these ranked query interfaces are auto filled with the system's query and the structured web pages are downloaded dynamically. These freshly downloaded structured web pages and the earlier downloaded web pages provided by object handler are now processed by ESDREI and the data is extracted as shown in Fig 6.18.

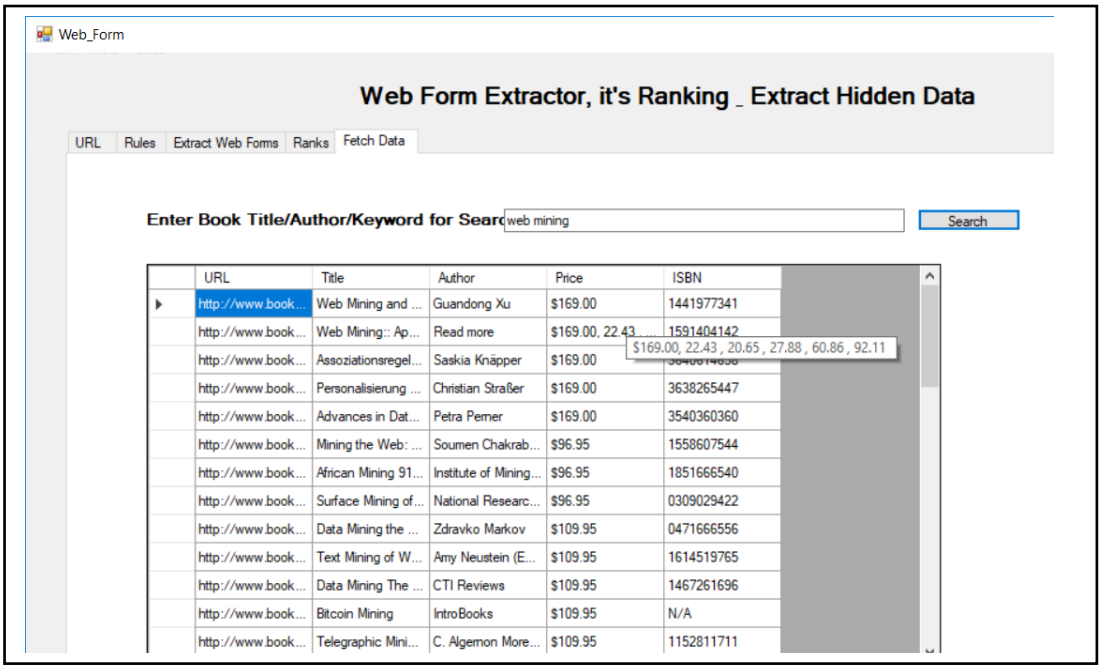

Figure 6. 18. Processed and Integrated Data from the Structured Web Pages

All the steps carried out so far are executed on the system side. The processing done on the end users side is shown next.

End User Side:

When the end user login into the system home page is shown to as shown in Fig 6.19. The home page has one text input box for writing the user query and a submit button which posts data supplied by the user to the system. For e.g. “web mining” query is issued by the user.

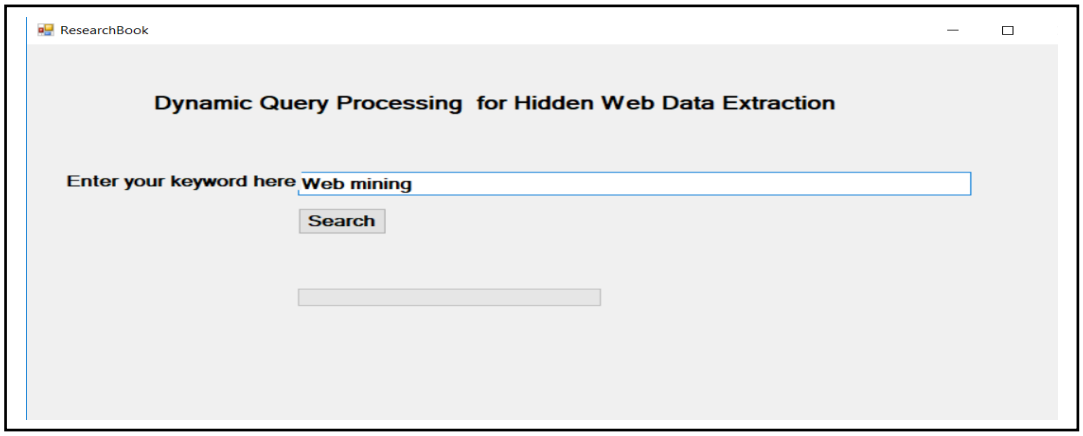

Figure 6. 19. Home Page of End User

When user query is fired, the results already processed and stored by DQPHDE are extracted and returned to the user. The downloaded unstructured web pages processed by CGRE and STMS are shown in first two tabs of the result screen as shown in Fig 6.20. The first tab of the result page shows the URL’s where only the abstract of the research papers are stored.

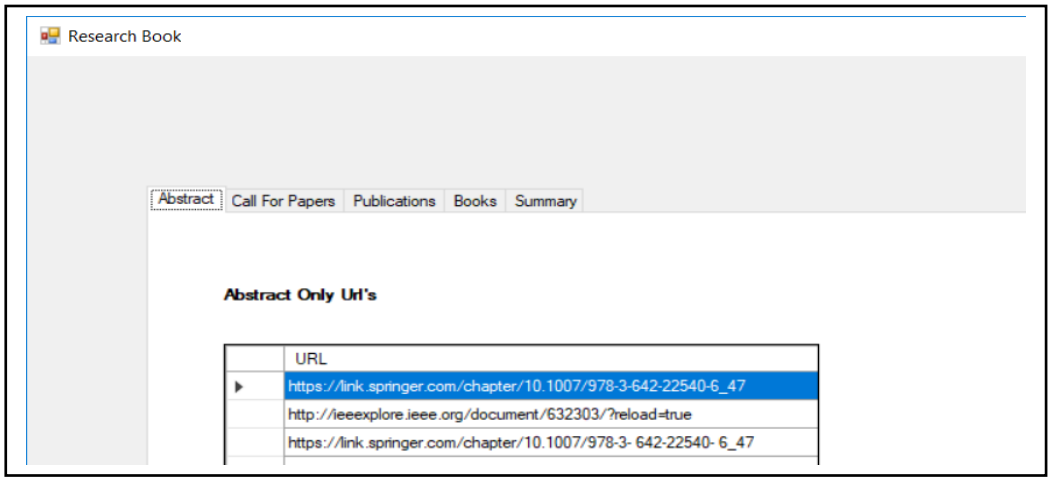

Figure 6. 20. The Links of Web Pages Containing the Abstracts

The second tab on the result screen shows the processed data of call for papers as shown in Fig 6.21.

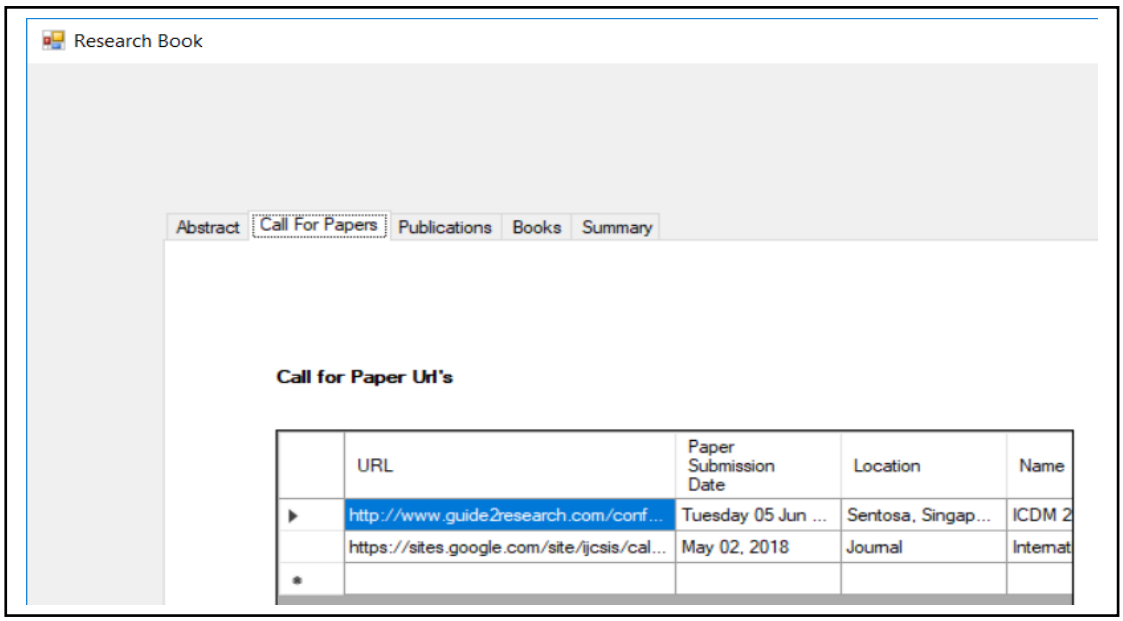

Figure 6. 21. The Call for Papers Data and Links of those URL's

The research articles are clustered using CGRE module of the proposed work and the results are shown in Fig 6.22. The user can group the research articles at run time also by clicking on his/her required key feature of clustering.

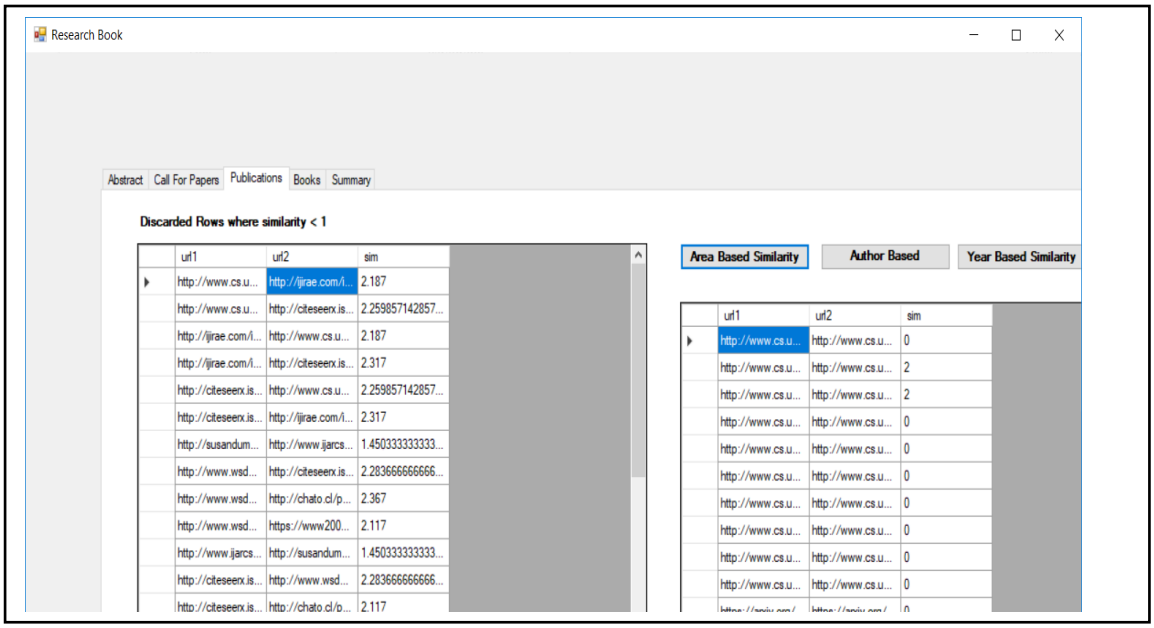

Figure 6. 22. Clustering of Research Articles

In the fourth tab the data from the hidden web is extracted by WFER and ESDREI module of the proposed work as shown in Fig 6.23. ESDREI extracts and integrates the data from multiple web pages. It removes the redundancy and combine the details of the entity which is scattered across various pages in WWW.

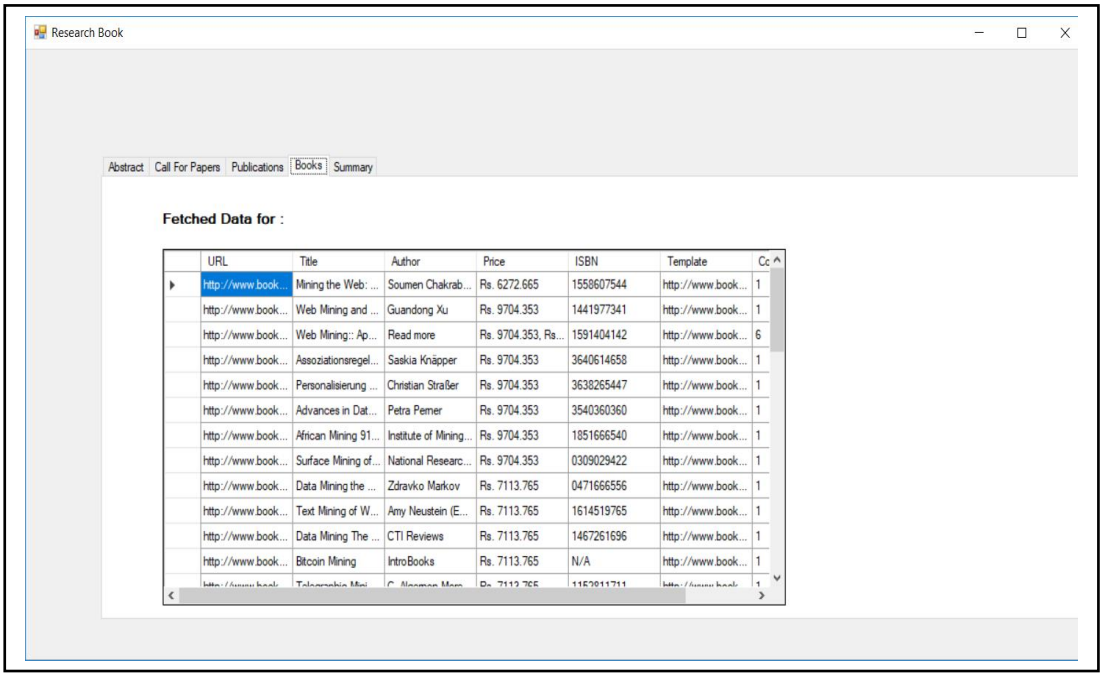

Figure 6. 23. Data Extracted from Structured Web Pages

The last tab on the user screen shows the processed data from unstructured web pages. This data is processed by STMS and integrated as well as summarized results are displayed to users as shown in Fig 6.24.

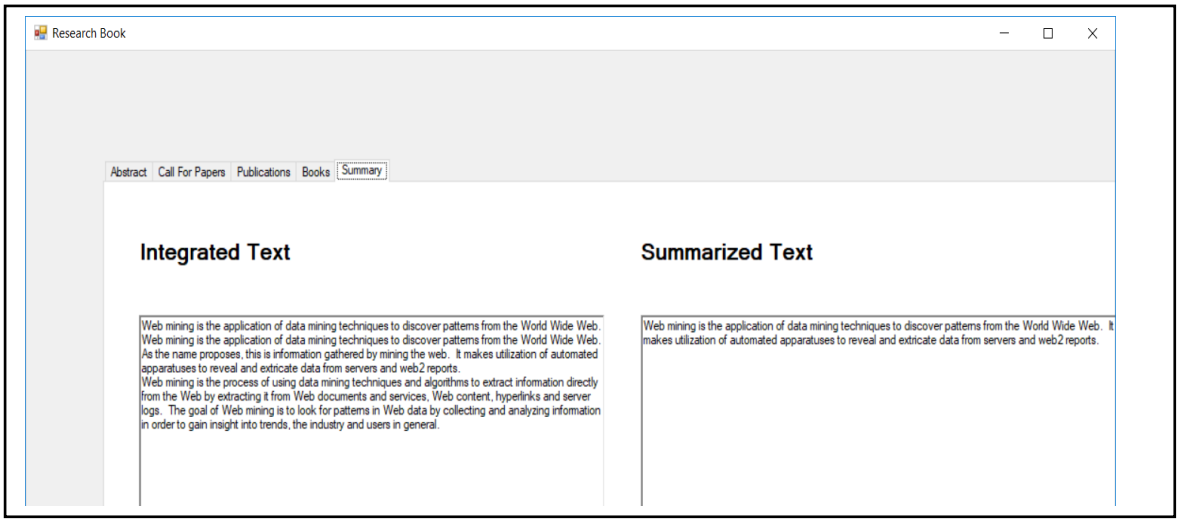

Figure 6. 24. The Integration and Summary of Unstructured Web Pages

6.4. DISCUSSION

The results of the proposed work have been compared with the already existing Academic Search Engines. The Academic Search Engines are created to meet the requirements of the researchers. When a researcher starts exploring a new area then they are not only interested in the research articles, they can also look for certain books relevant to their area. Then they can also go through the research articles which is provided by the academic search engines. Then after exploring the area they can be interested in call for paper details. User has to spend a lot of time on traditional search engines before accessing the research articles, in order to gather knowledge about the topic and search for books of a particular area. User also has to spend a lot of time on traditional search engines after accessing the research articles in order to gather information about the call for papers of a particular area. The current academic search engines meet only one requirement of the researchers which is providing the research articles. They ignore or do not even gather other information needed by them. So in the proposed work the hidden data related to academics is explored and an academic search engine is developed which fulfils all the requirements of the researchers. In order to fulfil the requirements, the proposed work DQPHDE extracts the data from both the surface web and the hidden web. By integrating the surface web information and hidden web information, a single portal is developed from where the user can get all the data required.

To compare the proposed work with the existing academic search engines, the queries are fired on these search engines and the limitations in their results are highlighted. For example in Fig 6.25 the user has fired the query “Data Mining” and the result page returned by Microsoft Academic is shown.

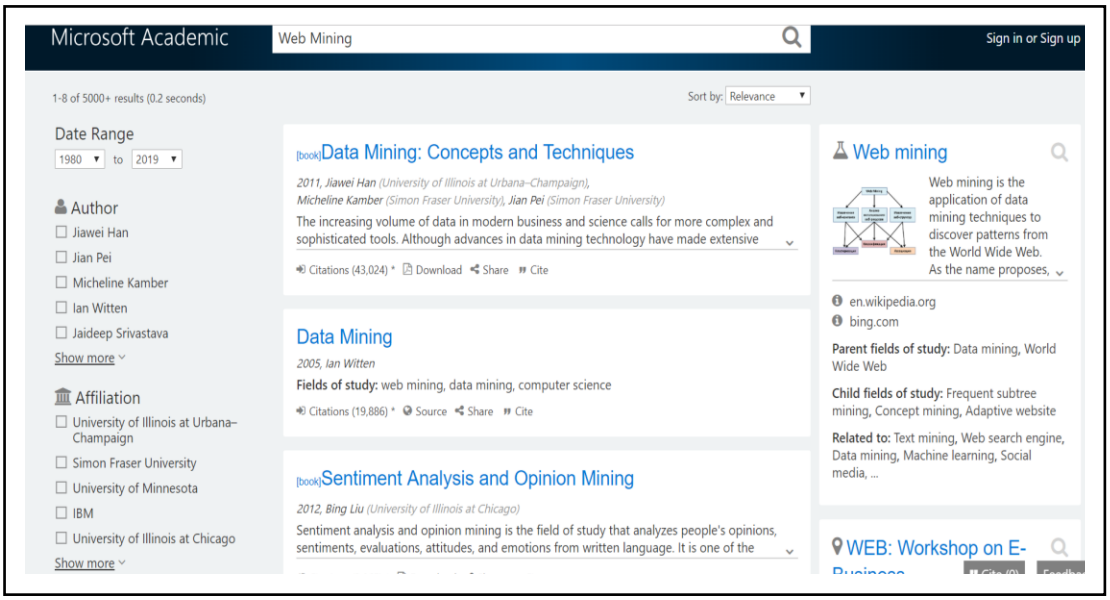

Figure 6. 25. Result Page of Microsoft Academic

The result page from Microsoft Academic search engine shows the research articles and the links from Wikipedia and Bing. Other details such as books details, summary and integrated information about the area, call for papers, journal details etc. needed by the researchers has not provided by Microsoft Academic which is provided by proposed work.

Another example of popular academic search engine is “Academia.edu”. When the query “Web Mining” is issued on Academia.edu the result page is shown in Fig 6.26

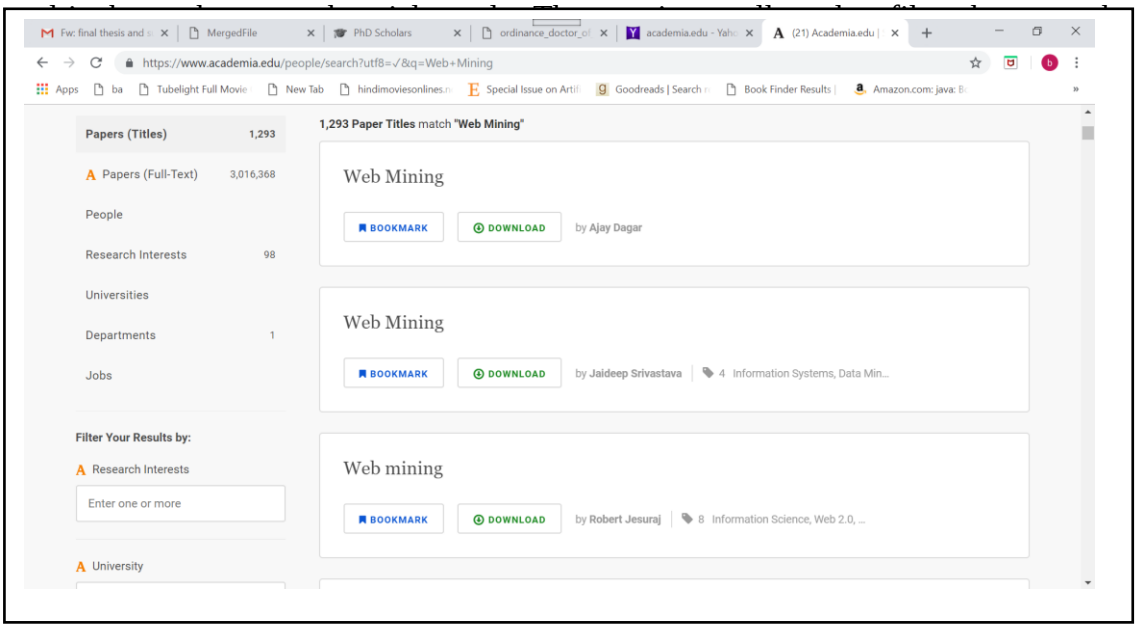

Figure 6. 26 Result Page of Academia.edu

Computing Research and Education (CORE) is another popular academic search engine. The query “Web Mining” is fired on CORE and the results returned are shown in Fig 6.27. The results returned by CORE contains the research articles. Here the results returned are more promising than the Academia.edu and Microsoft Academia. The results can be filtered by end user according to their requirements at run time but here also the research articles are provided to end users as results. The other information needed like book details, call for papers, and summary of articles etc., by the researchers is not provided here too.

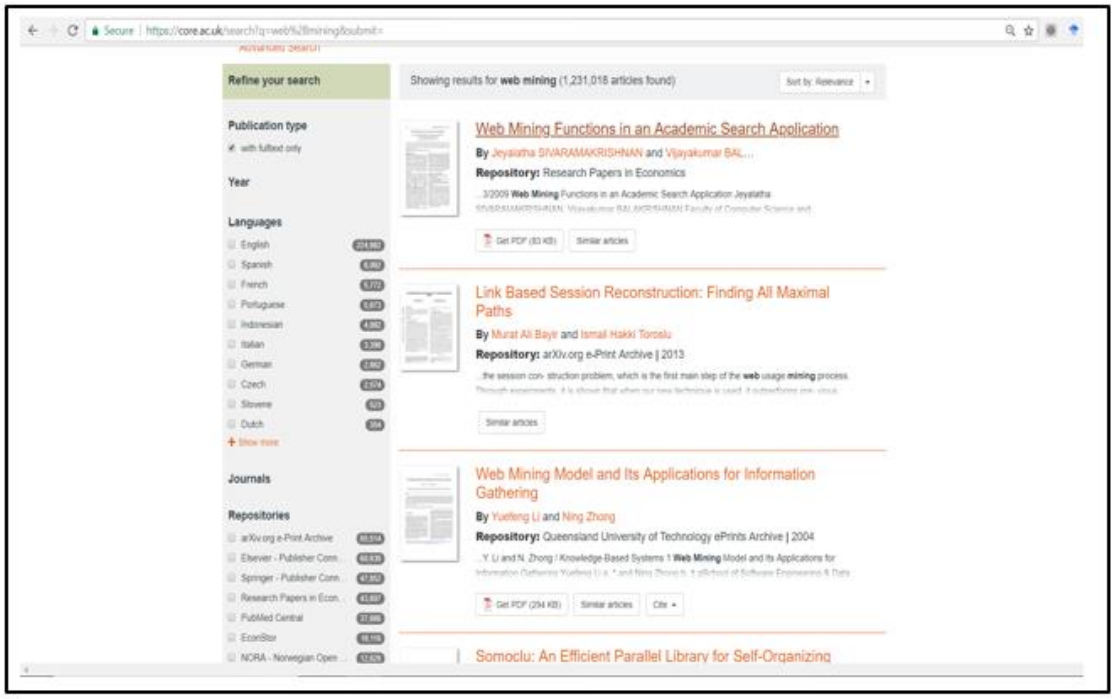

Figure 6. 27 Result Page of CORE

There are many other academic search engines like Directory of Open Access Journal (DOAJ), Bielefeld Academic Search Engine (BASE), Google Scholar that do not show all the information that user can expect as shown in Table 6.1.

Table 6. 1. Comparison of Different Academic Search Engines with the Proposed Academic Search Engine in this Research

	Microsoft Academic	Academia.edu	CORE	DOAJ	BASE	Google Scholar	Research Gate	DQPHDE
Call for Papers	No	No	No	No	No	No	No	Yes
Abstract	No	No	No	No	No	No	No	Yes
Books	No	No	No	No	No	No	No	Yes
Research Articles	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Summary	Links from Bing and Wikipedia	No	No	No	No	No	No	Yes
Integrated Tutorial	No	No	No	No	No	No	No	Yes

6.5. CONCLUSION

This chapter discussed the implementation details and the results of the proposed work DQPHWE. The research work decreased the user time to access the required information and increased the user satisfaction level by aggregating and processing the data from both the surface web and the hidden web. Aggregating both the surface and hidden web data for the researchers is a novel concept and this proposed work can be extended to different domains. The clustering of research articles is done by the weighted graph relationship approach. It also aggregates the other details such as call for papers, abstracts of research articles etc. for researchers that otherwise remained scattered across multiple web pages and the user has to go through multiple web pages to gather the information. It also fetches data from the hidden web. It aggregates the details of an entity scattered across several hidden web databases and successfully removes the duplicity of the records. The challenges faced by the researcher to access the desired information are successfully resolved by the DQPHDE and the results shows that it easily overcomes the limitations of the traditional academic search engines. Results are competitive and have an edge over the existing work.

CHAPTER 7

CONCLUSIONS AND FUTURE SCOPE

7.1. CONCLUSIONS

Search engines are the software programs by the help of which the end users find the desired web pages from WWW. The set of softwares which fall under this category are Google, Bing, and Yahoo etc. Users issue their query and the search engines look up in their index and find the list of document which contains the keywords found in user's query. The list of documents are then shown to the user as a result of their query. The problem is that the search engines are able to index only a part of the web called the surface web. The surface web is also known as visible web or publically indexable web (PIW). The crawlers are able to crawl the pages from surface web and hence the indexer is able to index the surface web only. A very large part of the WWW called as Hidden Web is not indexed by the search engines. Hence the hidden web pages are not shown to the user in result pages. The data in the hidden web is of high quality and quantity. End users themselves have to explore this part of WWW, which consumes a lot of time and effort. So this part of the WWW cannot be ignored and should be extracted and made available to the users directly just like the surface web.

The in-depth study of the literature highlighted that the size of the hidden web is increasing at a very fast pace but still less data has been extracted. A major part of the hidden web lies in the web server databases and this data is accessible by the help of the query interfaces. Extracting this data poses many problems. The first problem is the heterogeneous schema of the databases. This heterogeneous schema of databases hinders in the integration of data from multiple web databases. The second problem is that the traditional web crawlers are not able to process the web forms. Another problem encountered is that the users are interested in both the surface web and the hidden web but there is no source from which users can get this type of integrated information.

The above problems are solved by the proposed work Dynamic Query Processing for Hidden Web Data Extraction (DQPHDE). For the experimental purpose, the academic domain has been taken. The proposed prototype can be used by the researchers where they can find all kind of information about the domain of their interest. The system will reduce

the effort of users to extract the data from both the surface web and the hidden web. DQPHDE extracts all kind of web pages from WWW and processes them. Following modules have been proposed to obtain the solutions of the above discussed problems.

- **Web Document Clustering using Graph and Relationship Extraction (CGRE)**

In the beginning, the different types of web pages of a domain are downloaded and then they are classified as structured web pages, unstructured web pages, query interfaces etc. The predefined semantic classes store the basic structure of different types of web pages. With the help of these semantic classes, the web documents are classified. The different types of web documents are then forwarded to their respective object handlers. Each object handler processes its respective types of web pages in parallel. The main aim of this module i.e. CGRE is to cluster the research articles. It clusters the research articles using the linked relationship graph.

- **Semantic Based Text Mining and Summarization (STMS)**

STMS is an unstructured web document handler. The goal of STMS is to integrate and create a summary of text from multiple unstructured documents. It extracts all the data from unstructured document and processes it. The summarized and the integrated data is shown to the end users.

- **Web Form Extraction and Ranking (WFER)**

The objective of Web Form Extraction and Ranking (WFER) is to extract the web forms from the web pages and rank them. The web-accessible databases make up the lion's share of the hidden web [121], which are accessible through web forms only. So it is a major task to extract these web forms.

- **Expert System for Data Region Extraction and Semantic Integrated Graph of Named Entities (ESDREI)**

The goal of the module is to extract the data regions containing records from hidden web databases. The fuzzy rule based expert system has been proposed to extract the data region from result web pages. A knowledge base of facts and rules have been created to achieve this objective.

The Dynamic Query Processing for Hidden Web Data Extraction (DQPHDE) is implemented using the .NET Framework and SQL. The results show that the user can get the desired result from both the surface web and the hidden web using a single query. To extract the information about the entity, the end user explores both the surface web and the hidden web. It consumes a lot of time and effort of the user. The proposed system reduces the end users effort and saves time of the end users.

The proposed system creates a single portal where data from both the surface web and hidden web is summarized and integrated.

7.2. FUTURE SCOPE

Some of the issues that can be further explored or extended are given as follows:

- **Implementation of distributed and parallel hidden web crawler**

The size of WWW is very enormous and it is continuing to increase at a very fast pace. So the distributed and parallel crawling can be embedded in the system to make crawling procedure more effective.

- **Updation of hidden web pages**

The hidden web information is frequently updated. The frequency of updation can be calculated and the freshness of the data can be retained.

- **Covering more domains**

The web consists of information about different domains, the work can be extended to work on multiple domains and can provide integrated data for those domains.

- **Improvement of storage mechanism**

The size of data on WWW is very large. The traditional storage mechanism slow the searching mechanism. So the different kind of big data storage tools can be used such as Apache Hadoop, NoSQL, and Hive etc.

- **Improved Ranking of Research Articles**

The research articles can be read by the readers and after reading the articles, the end user can comment on the articles. Based on the likes and the analysis of the comments, the ranking of the research articles done. It will add user perspective in ranking the research articles.

There are still a lot of challenges in hidden web data extraction but with the different interests of researchers and organization in hidden web and academic search engine domains, there is no doubt that in future new worthy answers will be provided which will help in better extraction of data from hidden web.

REFERENCES

[1] https://www.internetworldstats.com/stats.htm

[2] http://www.internetlivestats.com/one-second/

[3] Eduard c. Dragut, Weiyi Meng and Clement T. Yu, "Deep web query interface understanding and integration", Synthesis Lectures on Data Management #26, June 2012, ISBN 9781608458950.

[4] Seymour, T., Frantsvog, D., & Kumar, S.,"History of search engines", International Journal of Management & Information Systems, 2011 15(4), 47-58, ISSN 2157-9628.

[5] https://en.wikipedia.org/wiki/Academia.edu.

[6] Dutta, M., & Bansal, K. L, "A Review Paper on Various Search Engines (Google, Yahoo, Altavista, Ask and Bing)", International Journal on Recent and Innovation Trends in Computing and Communication, 2016, ISSN 2321-8169.

[7] Vengateshwaran, M., & EVR, M. K. "Web Mining Research Direction and Open Source Tools", International Journal of Advanced Research in Computer Science and Software Engineering, 2014, ISSN 2277-6451.

[8] Botluk, D.," Features – Search Engine Comparison Chart", 2004, Article.

[9] Knight, D., Holt, A., & Warren, J., "Search engines: a study of nine search engines in four categories", Journal of Health Informatics in Developing Countries, 2009, ISSN 1178-4407.

[10] http://www.cs.cornell.edu/Info/Department/Annual95/Faculty/Salton.html.

[11] https://en.wikipedia.org/wiki/Gerard_Salton#cite_note-father-IR-1.

[12]http://www.addondashboard.com/Article/The-first-search-engines--shy--rchieVeronica-Gopher-and-Wandex/197283.

[13] http://www.hartnell.cc.ca.us/faculty/jlagier/internet/gopher.html.

[14] http://www.robotstxt.org/db/wanderer.html.

[15] http://www.abcseo.com/seo-book/alta-vista.htm

[16] en.wikipedia.org/wiki/Google

[17] "Infrastructure Services for Open Access". Infrastructure Services for Open Access C.I.C. Retrieved 2013-03-05.

[18] https://doaj.org/about

[19] http://www.core.edu.au/

[20] https://www.base-search.net/about/en/

[21] https://en.wikipedia.org/wiki/Google_Scholar

[22] Microsoft. "Academic Knowledge API". Retrieved 29 January 2017.

[23] ResearhGate.net

[24] Academai.edu

[25] Michael k. Bergman, "The deep web: surfacing hidden value", Bright Planet Corp, 2001.

[26] B. He, M. Patel, Z. Zhang, and K. C.-C. Chang, "Accessing the deep web", Communications of the ACM, 50, May 1, 2007.

[27] Kunder, Maurice de, "Estimating search engine index size variability: a 9-year longitudinal study", Springer, 2016.

[28] Mike Burner, "Crawling towards Eternity: Building an archive of the World Wide Web", Web Techniques Magazine, May 1997

[29] Carlos Castillo, "Effective Web Crawling", Ph.D Thesis, Department of Computer Science, University of Chile, November 2004.

[30] S. Raghavan and H. Garcia-Molina, "Crawling the Hidden Web", In Proceedings of 27th International Conference on Very Large Data Bases, Roma, Italy, pp. 129–138, 2001.

[31] Dilip Kumar Sharma and A. K. Sharma, "Deep Web Information Retrieval Process: A Technical Survey", In International Journal of Information Technology & Web Engineering, USA, Vol 5, No. 1, pp.1-22, 2010, ISSN: 1554-1045.

[32]J. Akilandeswari and N. P. Gopalan, "An Architectural Framework of a Crawler for Locating Deep Web Repositories Using Learning Multi-Agent Systems", In Proceedings of the 2008 Third International Conference on Internet & Web Applications and Services, PP.558-562, 2008.

[33] Dilip Kumar Sharma and A. K. Sharma, “Search Engine: A Backbone for Information Extraction in ICT Scenario”, In International Journal of ICTHD, USA , Vol. 3, No. 2, pp. 38-51, 2011, ISSN: 1935-5661.

[34] Brian Pinkerton, “WebCrawler: Finding What People Want”, PhD Thesis, University of Washington, 2000.

[35] S. Chakrabarti, M. Berg, and B. Dom, “Focused Crawling: A New Approach to Topic-Specific Web Resource Discovery”, In Proceedings of the eighth international conference on World Wide Web, 1999

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

[37] D. Yadav, A.K. Sharma, and J.P. Gupta, "Parallel Crawler Architecture and Web Page Change Detection Techniques". In WSEAS Transaction on Computers, Issue 7, Vol 7, PP. 929-941, ISSN: 1109-2750, 2008.

[38] J. Hammer and J. Fiedler, “Using Mobile Crawlers to Search the Web Efficiently”, In International Journal of Computer and Information Science, pp.36-58, 2000, ISSN 1708-0460.

[39] P. Boldi, B. Codenott, M. Santini, and S. Vigna, "UbiCrawler: A Scalable Fully Distributed Web Crawler”, In Software: Practice & Experience, Vol. 34, No. 8, 2004.

[40] Chang, K., He, B., Li, C., Patel, M., Zhang, Z,”Structured databases on the Web: Observations and implications”, ACM SIGMOD, 2004.

[41] Madhavan, J., Cohen, S., Dong, X.L., Halevy, A.Y., Jeffery, S.R., Ko, D., Yu, C, “Web scale data integration: You can afford to pay as you go”, In CIDR, 2007.

[42] Crescenzi, V., Mecca, G., Merialdo, P RoadRunner, “Towards automatic data extraction from largeWeb sites”, In Proceedings of the 27th International Conference on Very Large Data Bases (VLDB 2001), pp. 109–118 ,2001.

[43] Doorenbos, R., Etzioni, O., Weld, D, “A scalable comparison-shopping agent for the World-Wide Web”, In Proceedings of the First International Conference on Autonomous Agents,1997.

[44] Kushmerick, N., Weld, D., Doorenbos, R, “Wrapper induction for information extraction”, In Proceedings of the 15th International Joint Conference on Artificial Intelligence (IJCAI 1997), pp. 729–737,1997.

[45] Lerman, K., Minton, S., Knoblock, C, "Wrapper maintenance: A machine learning approach", Journal of Artificial Intelligence Research (JAIR), 2003, ISSN 1076 - 9757.

[46] Raghavan, S., Garcia-Molina, H, "Crawling the hidden Web", In Proceedings of 27th International Conference on Very Large Data Bases (VLDB 2001), pp. 129–138, 2001.

[47] Arasu, A., Garcia-Molina, H, "Extracting structured data from Web pages", In Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data (SIGMOD 2003), pp. 337–348, 2003.

[48] Barbosa, L., Freire, J, "Searching for hidden-Web databases", In Proceedings of the 8th ACM SIGMOD International Workshop on Web and Databases (WebDB 2005), pp. 1–6, 2005.

[49] He, B., Chang, K, "Statistical schema matching across Web query interfaces", In Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data (SIGMOD 2003), pp. 217–228, 2003.

[50] He, H., Meng, W., Yu, C., Wu, Z, "WISE-Integrator: An automatic integrator of Web search interfaces for e-commerce", In VLDB 2003. LNCS, vol. 2944, pp. 357–368. Springer, Heidelberg, 2004.

[51] Madhavan, J., Cohen, S., Dong, X.L., Halevy, A.Y., Jeffery, S.R., Ko, D., Yu, C, "Webscale data integration: You can afford to pay as you go", In CIDR, 2007.

[52] McCann, R., AlShelbi, B., Le, Q., Nguyen, H., Vu, L., Doan A, "Maveric: Mapping maintenance for data integration systems", In Proceedings of the 31st International Conference on Very Large Data Bases, 2005.

[53] Wang, J., Wen, J., Lochovsky, F., Ma, W, "Instance-based schema matching for Web databases by domain-specific query probing", In Proceedings of the 30th International Conference on Very Large Data Bases, 2004.

[54] Wu, W., Yu, C., Doan, A., Meng, W, "An interactive clustering-based approach to integrating source query interfaces on the Deep Web", In SIGMOD 2004.

[55] Wu, W., Doan, A., Yu, C, "Merging interface schemas on the Deep Web via clustering aggregation", In Proceedings of the 5th IEEE International Conference on Data Mining (ICDM 2005), pp. 801–804, 2005.

[56] Deepak Kumar, Dhanesh kumar, Dr. Mahesh Kumar, "Integration of Search Interfaces in Deep Web Databases for Specific Domains", IOSR Journal of Computer Engineering (IOSR-JCE) ISSN: 2278-8727, Volume 17, Issue 4, Ver. IV, 2015, pp 75-90.

[57] Radhouane Boughammoura, Lobna Hlaoua, "VIQI: A New Approach for Visual Interpretation of Deep Web Query Interfaces", Published in 2012 International Conference on Information Technology and e-Services.

[58] M. Ester, H-P Kriegel, J. Sander, X. Xu, ''A density-based algorithm for discovering clusters in large spatial databases with noise'', In proceeding of KDD-96, 1996.

[59] Weifeng Su, Hejun Wu, "Understanding Query Interfaces by Statistical Parsing", ACM Transactions on the Web, Vol. 7, No. 2, Article 8, May 2013.

[60] Tian Zhao 1,*, Chuanrong Zhang 2 and Weidong Li 2, "Adaptive and Optimized RDF Query Interface for Distributed WFS Data", International Journal of Geo-Information, 2017, ISSN 2220-9964.

[61] Wensheng Wu, AnHai Doan, Clement Yu, and Weiyi Meng, "Modeling and Extracting Deep-Web Query Interfaces", Advances in Information & Intelligent Sys, 2009.

[62] Grishman, Ralph; Sundheim, B., "Message Understanding Conference A Brief History", In Proc. International Conference on Computational Linguistics, 1996.

[63] J. Guo, G. Xu, X. Cheng, and H. Li, "Named entity recognition in query", SIGIR'09.

[64] M. J. Cafarella, D. Downey, S. Soderland, and O. Etzioni, "Know it now: Fast, scalable information extraction from the web", In EMNLP, 2005.

[65] D. Nadeau, "Semi-supervised named entity recognition: learning to recognize 100 entity types with little supervision", PhD thesis, 2007.

[66] Zaiqing Nie, Ji-Rong Wen, Wei-Ying Ma, "Statistical Entity Extraction From the Web", Proceedings of the IEEE, Volume: 100, Issue 9, Sept. 2012, pp.2675-2687.

[67] Peter Exner and Pierre Nugues "Entity extraction: From unstructured text to DBpedia RDF triples", The Web of Linked Entities Workshop In Proceedings of the Web of Linked Entities Workshop in conjunction with the 11th International Semantic Web Conference (ISWC 2012) , 2012

[68] Nathanael Chambers, James Allen, and Lucian Galescu ,"Using Semantics to Identify Web Objects", Proceedings of the National Conference on Artificial Intelligence , American Association for Artificial Intelligence, 2006.

[69] Ke Zhai, Zornitsa Kozareva, Yuening Hu, Qi Li and Weiwei Guo," Query to Knowledge: Unsupervised Entity Extraction from Shopping Queries using Adaptor Grammars" SIGIR '16 Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval Pisa, Italy — July 17 - 21, 2016.

[70] Andreas Eiselt, Alejandro Figueroa, "A Two-Step Named Entity Recognizer for Open-Domain Search Queries", Proceedings of the 23rd International Conference on Computational Linguistics, Beijing, August 2010.

[71] Xiaoxin Yi,Sarthak Shah, "Building taxonomy of web search intents for name entity queries", WWW '10 Proceedings of the 19th international conference on World wide web Raleigh, USA, 2010.

[72] B. He and K. C.-C. Chang, "Statistical Schema Matching across Web Query Interfaces", In SIGMOD, 2003.

[73] A. D. Sarma, X. Dong, and A. Halevy, "Bootstrapping pay-as-you-go data integration systems", In SIGMOD, 2008.

[74] Disheng Qiu, Luciano Barbosa, Xin Luna Dong, "DEXTER: Large-Scale Discovery and Extraction of Product Specifications on the Web", Proceedings of the VLDB Endowment, Vol. 8, No. 13, 2015.

[75] Manuel Álvarez, Juan Raposo, Fidel Cacheda and Alberto Pan. "A Task-specific Approach for Crawling the Deep Web", Engineering Letters, EL_13_2_19 (Advance online publication), 4 August 2006.

[76] S.Raghavan and H. Garcia-Molina. "Crawling the hidden web". VLDB, 2001.

[77] Rosy Madaan, Ashutosh Dixit, A.K. Sharma and Komal Kumar Bhatia, "A Framework for Incremental Hidden Web Crawler", International Journal on Computer Science and Engineering Vol. 02, No. 03, 2010, ISSN 2347-2693.

[78] M. W. Berry, Proceedings of the Third SIAM International Conference on Data Mining (SDM-2003) Workshop on Text Mining, San Francisco, CA, May 2003.

[79] M. Grobelnik, Proceedings of IEEE International Conference on Data Mining (ICDM- 2001) Workshop on Text Mining (TextDM'2001), San Jose, CA, 2001.

[80] M. Grobelnik, Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence(IJCAI-2003) Workshop on Text Mining and Link Analysis (TextLink-2003), Acapulco, Mexico, Aug. 2003.

[81] M. A. Hearst "Untangling text data mining", In Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics (ACL-99), pages 3–10, College Park, MD, June 1999.

[82] M. A. Hearst, "What is text mining", Oct. 2003.

[83] D. Mladeni, Proceedings of the Sixth International Conference on Knowledge Discovery and Data Mining (KDD-2000) Workshop on Text Mining, Boston, MA, Aug. 2000.

[84] M.F. Porter, "An Algorithm for Suffix Stripping, Program", vol. 14, no. 3, pp. 130-137, 1980.

[85] Wu, V. Kumar, J.R. Quinlan, J. Ghosh, Q. Yang, H. Motoda, "Top 10 Algorithms in Data Mining", Knowledge Information Systems, vol. 14, no. 1, pp. 1 37, 2007.

[86] Han, Jiawei, Micheline Kamber, and Jian Pei, Data mining, south East Asia edition: Concepts and techniques, Morgan kaufmann, 2006.

[87] Kesavaraj, G., and S. Sukumaran. "A study on classification techniques in data mining." In 2013 Fourth International Conference on Computing, Communications and Networking Technologies (ICCCNT), pp. 1-7. IEEE, 2013.

[88] A. K. Jain, M. N. Murty and P. J. Flynn, "Data Clustering: A review", ACM Computing Surveys, vol. 31, no. 3, 1999.

[89] Chen, C., Tseng, F. S., & Liang, T., "An integration of WordNet and fuzzy association rule mining for multi-label document clustering", Data & Knowledge Engineering, 2010.

[90] G. Hamerly, "Learning Structure and Concepts in Data using Data Clustering", PhD Thesis. University of California, San Diego, 2003.

[91] E. Forgy, "Cluster Analysis of Multivariate Data: Efficiency versus Interpretability of Classification", *Biometrics*, vol. 21, pp. 768-769, 1965.

[92] J. Bezdek," A Convergence Theorem for the Fuzzy ISODATA Clustering Algorithms", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 2, pp. 1-8, 1980.

[93] G. McLachlan and T. Krishnan, "The EM algorithm and Extensions", John Wiley & Sons, Inc., 1997.

[94] H. Frigui and R. Krishnapuram, "Clustering by Competitive Agglomeration. Pattern Recognition Letters", vol. 30, no. 7, pp. 1109-1119, 1997.

[95] S. C. Johnson, "Hierarchical Clustering Schemes", Psychometrika, 1967.

[96] P. Sneath and R. Sokal,"Numerical Taxonomy", Freeman, London, UK, 1973.

[97] M. Anderberg," Cluster Analysis for Applications", Academic Press, New York, USA, 1973.

[98] Mr. Rahul Patel, Mr. Gaurav Sharma, "A survey on text mining techniques", International Journal Of Engineering and Computer Science, ISSN 2319-7242, Vol 3 Issue 5, May 2014, pp.5621-5625

[99] S. Bandyopadhyay and S. Saha, "Unsupervised Classification: Similarity Measures, Classical and Metaheuristic Approaches, and Applications", Springer, 2013.

[100] Sumit Ganguly and Gaurav Veda, "A new randomized algorithm for Document Fingerprinting", Indian Institute of Technology, 2000.

[101] Dittmann Jana, Behr Alexander, Stabenau Mark, Schmitt Peter, Schwenk Jörg, Ueberberg Johannes, "Combining digital Watermarks and collusion secure Fingerprints for digital Images", German National Research Center for Information Technology, Darmstadt, Germany, 1999.

[102] Schleimer Saul, Wilkerson Daniel S., Aiken Alex, "Winnowing: Local Algorithms for Document Fingerprinting", in Proceedings of the 2003 ACM SIGMOD international conference on Management of data, NY, USA, 2003.

[103] Xiao, Y., Tao, Y., Li, Q, "Web page adaptation for mobile device", In Wireless Communications, Networking and Mobile Computing, WiCOM '08. 4th International Conference, 2008.

[104] Cai, D., Yu, S., Wen, J.R., Ma, W.Y, "Extracting content structure for web pages based on visual representation", In APWeb 2003. LNCS, Springer 2003.

[105] Saad, M.B., Gancarski, S, "Using visual pages analysis for optimizing web archiving", In Proceedings of the 2010 EDBT/ICDT Workshops, ACM (2010).

[106] Fariza Fauzi, Jer-Lang Hong, and Mohammed Belkhatir, "Webpage segmentation for extracting images and their surrounding contextual information", In MM '09: Proceedings of the seventeen ACM international conference on Multimedia, 2009.

[107] Diao, Y., Lu, H., Chen, S., and Tian, Z., "Toward Learning Based Web Query Processing", In Proceedings of International Conference on Very Large Databases, 2000.

[108] Lin, S.-H. and Ho, J.-M., "Discovering Informative Content Blocks from Web Documents", In Proceedings of ACM SIGKDD'02, 2002.

[109] Kaasinen, E., Aaltonen, M., Kolari, J., Melakoski, S., and Laakko, T., "Two Approaches to Bringing Internet Services to WAP Devices", In Proceedings of 9th International World-Wide Web Conference, 2000, pp. 231-246.

[110] Buyukkokten, O., Garcia-Molina, H., and Paepche, A., "Accordion Summarization for End-Game Browsing on PDAs and Cellular Phones", In Proceedings of the Conference on Human Factors in Computing Systems, CHI'01, 2001.

[111] Embley, D. W., Jiang, Y., and Ng, Y.-K., "Record-boundary discovery in Web documents", In Proceedings of the 1999 ACM SIGMOD international conference on Management of data, Philadelphia PA, 1999, pp. 467-478.

[112] Buttler, D., Liu, L., and Pu, C., "A Fully Automated Object Extraction System for the World Wide Web", In International Conference on Distributed Computing Systems, 2001.

[113] Deng Cai, Shipeng Yu,Ji-Rong Wen and Wei-Ying Ma, "VIPS: A Vision based Page Segmentation Algorithm", Technical Report MSR-TR-2003-79, Microsoft Technical Report, 2003.

[114] Iqbal, R., Doctor, F., More, B., Mahmud, S., & Yousuf, U, "Big data analytics: computational intelligence techniques and application areas", International Journal of Information Management, 2016, ISSN 0268-4012.

[115] Elaine Rich & Kevin Knight, Artificial Intelligence. Second Edition. Tata McGraw Hill Edition, 1991.

[116] Maitri Patel, Paresh Virparia , Dharmendra Patel, " Web based Fuzzy Expert System and Its Applications – a Survey", International Journal of Applied Information Systems (IJAIS) Foundation of Computer Science FCS, New York, USA Volume 1– No.7, March 2012, ISSN 2249-0868.

[117] Niwa, K., Sasaki, K. and Ihara, H., "An Experimental Comparison of Knowledge Representation Schemes, in Principles of Expert Systems", IEEE Press, New York (pp. 133–140), 1998.

[118] Zadeh, L.A, "Fuzzy Sets. Information and Control", 8, 1965.

[119] Schneider, M., Langholz, G., Kandel, A. and Chew, G, "Fuzzy Expert System Tools", John Wiley & Sons, USA, 1996.

[120] https://www.popsci.com/dark-web-revealed

[121] Chris Sherman and Gary Price, "Uncovering Information Sources Search Engines Can't See", CyberAge Books, November 2001.

[122] https://www.deepweb-sites.com/

[123] https://www.w3.org/TR/owl-xmlsyntax/apd-example.html

[124] https://en.wikipedia.org/wiki/BASE_(search_engine)

BRIEF PROFILE OF RESEARCH SCHOLAR

Babita Ahuja did her M.Tech. (Computer Science and Engineering) from Maharishi Dayanand University, Rohtak in 2007 and B.Tech. (Computer Science) from Maharishi Dayanand University, Rohtak in 2004. Ms. Babita has over 12 years of experience in teaching B.Tech. and M.Tech. courses. Her areas of interest include Operating System, Semantic Web, Web Technologies, Search Engines and Hidden Web. She has published 13 research papers in various journals and conferences of international fame. She has worked as Assistant Professor in the department of Computer Science & Engineering at Manav Rachna University, Faridabad for 11 years and as a Lecturer in Computer Science Engineering in C.I.T.M for more than 1 year.

LIST OF PUBLICATIONS OUT OF THESIS

List of Published Papers

S. No	Title of Paper	Name of Journal where published	No.	Volume & Issue	Year	Pages
1	Expert system for web form extraction and ranking of web forms (SCIE)	Journal of Engineering Technology		6	2018	238-253
2	Hidden Data extraction Using URL Templates Processing (SCOPUS)	Advances in Intelligent Systems and Computing, Springer		654	2017	111-126
3	Hidden Web Extractor Dynamic Way to Uncover The Deep Web (UGC Approved)	International Journal on Computer Science and Engineering	6	4	2012	1137-1145
4	Expert System for Data Region Extraction and Semantic Integrated Graph of Named Entities (UGC Approved)	International Journal of Electronics Engineering	1	9	2017	84-91
5	Surfacing and Ranking the Deep Web (UGC Approved)	Journal of Network Communications and Emerging Technologies	2	5	2015	175-178
6	Hidden Web Data Extraction Tools	International Journal of Computer Applications	15	82	2013	9-15
7	SCUM : A Hidden Web Page Ranking Technique	International Journal of Innovative Research in Advanced Engineering		10	2014	162-167

List of Papers in National/International Conference

8. Ahuja, B., Pillai, A., & Juneja, D, "Dynamic Query Processing for Hidden Web Data Extraction INDIACom 2015, 9th International Conference on Computing for sustainable Global Development", IEEE sponsored scheduled on 11-13 march 2015 (IEEE sponsored)

9. Ahuja, B., Pillai, A., & Juneja, D, "Hidden Data extraction Using URL Templates Processing, CSI-2015, 50th Golden Jubilee Annual Convention on 'Digital Life' held at New Delhi (2-5 December). (SCOPUS INDEXED)

10. Ahuja, B., Pillai, A., & Juneja, D, "Surfacing and Ranking the Deep Web", International Conference on Recent Trends in Computer and Information Technology Research, 25-26 September 2015.

11. Ahuja, B., and Pillai, A, "Review of Web Entity Extraction Techniques", National Conference on "New Horizons in Technology for Sustainable Energy and Environment (NHTSEE 2017).