Dynamic Query Processing for Hidden Web Data Extraction From Academic Domain

Babita Ahuja[1], Anuradha Pillai[2], Deepika Punj[2,*] and Jyoti Verma[2]

[1]*MRCE, Faridabad, India*
[2]*J.C. Bose University of Science and Technology, YMCA, Faridabad, India*
E-mail: babitaspark@gmail.com; anuangra@yahoo,com; deepikapunj@gmail.com; justjyoti.verma@gmail.com
[*]*Corresponding Author*

Received 30 July 2020; Accepted 04 September 2020;
Publication 16 December 2020

Abstract

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 and hence they can be displayed to users as per their input query. In contrast to this, hidden web documents are neither indexable nor crawlable by the traditional search engines due to disconnected URL's, no-index tag, user authentication, web form processing. Also, since the information is scattered across multiple web pages, users find it difficult to hop between multiple pages to find the desired information. Hence, there is dire need of hidden web crawlers which could extract the data from hidden web databases and uncover this big part of WWW. In this research, a novel framework "Dynamic Query Processing for Hidden Web Data Extraction (DQPHDE)" has been proposed to extract such hidden web data and integrate it with the data from surface web to meet user's requirements. DQPHDE makes use of clustering, semantic based text mining and fuzzy rule based system to carry out the desired task. The results of the proposed work were compared with the existing academic search engines like 'Microsoft Academic' and 'Academia.edu' etc, and our proposed work

Journal of Web Engineering, Vol. 19_7–8, 931–970.
doi: 10.13052/jwe1540-9589.19782

outperforms them in fetching the information and then integrating the related information for other pages.

Keywords: Surface web, hidden web, dynamic query processing, text summarization, semantic fuzzy rules.

1 Introduction

The World Wide Web has become the largest source of digital information [1]. It has diverse range of information about studies, fashion, politics, tourism, social networking, mail systems, vehicles, business, sports, cooking, countries, history, illegal activities, and drugs and so on. 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 [2]. According to Internet Live Stats [3], 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 to find the desired information from this bulk information various search engines have been developed like Google, Bing, yahoo, Ask.com, AOL.com etc. There are different types of search engines. Some search engines are designed to extract the unstructured data from WWW while others extract the structured data [4]. The unstructured web documents are the HTML static web pages which are stored on the web servers. This type of web is known as surface web. On other hand, most of the structured web pages are the dynamic web pages that are stored in databases and they are generated only when user query is issued using the query interfaces. This kind of web is known as hidden web [5, 7].

The data in the web server databases is of very high quality and quantity. These query interfaces acts as a powerful tool for the human users. The humans can easily process them and access the data behind the query interfaces. Many tools have been developed for hidden web data extraction [6] from different domains such as people information, government federal information, geological surveys, medications, government documents, academic research documents etc. In this research the academic research documents have been taken. There are a lot of popular digital academic search engines such as DOAJ (Directory of Open Access Journals) [8], CORE (Computing Research and Education) [9], BASE (Bielefeld Academic Search Engine) [10], Google Scholar [11], Microsoft Academic [12], Research

Gate [13] and Academia.edu [14] etc. The idea is to organize and ease the information retrieval process for the researcher.

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 result pages. Thus there is a need for a system which processes the surface web as well as hidden web data automatically and returns integrated results to the user. To achieve this goal, a novel approach "Dynamic Query Processing for Hidden Web Data Extraction" (DQPHDE) is being proposed in this research that exploits the artificial intelligence technique. It extracts the data behind the query interfaces, processes and then integrates the information from different sections of WWW.

2 Related Work

Hidden Data Integrators accesses the hidden web content where the data of a domain from multiple sites is extracted and integrated. This 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 [15, 16]. The input form is then analysed and the domain of the form is identified. Now, the input form and the mediator schema get mapped. 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.

A system Dexter [17] has been developed which extracts the websites having product details and extracts the products detail from them. 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, In-Site Crawling, Specification Detection, and Specification-Extraction.

Deepbot [18] was a hidden web crawler developed for both client side as well as server side. For handling the client side data, client side scripting

and session management is handled. Conventional crawlers were not able to handle both these things. 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: Route Manager, Configuration Manager, Download Manager, Content Manager, Data repository, Indexer, Searcher.

The Directory of Open Access Journals (DOAJ) [8] is a website that lists open access journals and is maintained by Infrastructure Services for Open Access (IS4OA) [9]. The project defines open access journals as scientific and scholarly journals that meet high quality standards by exercising peer review or editorial quality control and "use a funding model that does not charge readers or their institutions for access". 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, the knowledge is free here.

CORE (Computing Research and Education): CORE is an association of university departments of computer science in Australia and New Zealand. Prior to 2004, it was known as the Computer Science Association, CSA [10]. 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 aggregating the world's open access research papers. When the user issue 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 the similar articles.

BASE (Bielefeld Academic Search Engine): BASE is a multi-disciplinary search engine for scholarly internet resources, created by Bielefeld University Library in Bielefeld, Germany. It is based on free and open-source software such as Apache Solr and VuFind [11]. It harvests Open Archive Initiative metadata from institutional repositories and other academic digital libraries that implement the Open Archives Initiative Protocol for Metadata Harvesting (OAI-PMH), and then normalizes and indexes the data for searching. In addition to OAI metadata, the library indexes selected web sites and local data collections, all of which can be searched via a single search interface. Users can search bibliographic metadata including abstracts, if available. However, BASE does not currently offer full text search.

Google Scholar: Another Google search engine, but quite different from its prime engine, Google Scholar scans for a wide range of academic literature. The search results are extracted from university repositories, online journals, and other related web sources [12]. Google Scholar helps researchers find sources that exist on the internet. User can customize his/her search results to a particular field of interest, region, or institution, for example 'psychology, Harvard University.' This will give access to relevant documents.

Microsoft Academic: Microsoft is a free public search engine for academic publications and literature, developed by Microsoft Research. Relaunched in 2016, the tool features an entirely new data structure and search engine using semantic search technologies. It currently indexes over 375 million entities, 170 million of which are academic papers [13]. The Academic Knowledge API offers information retrieval from the underlying database using rest endpoints for advanced research purposes.

Research Gate: Research gate was launched in 2008. Research gate include 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 [14].

Academia.edu: It is another academic social networking site launched in 2008. It has many features such as maintaining list of one's publication, following topics or researchers with similar interest and publishing blog posts [19]. It also provides users with the information on how many people visited their profile from which countries and what they found interesting on the user profile.

The comparison between the different digital academic libraries is shown in Table 1.

3 Proposed Work

This section presents 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 involves all kinds of web documents requires clustering, text mining, summarization, query interface detection and data region extraction from web documents. It also provides a single platform where all kinds of information such as

Table 1 Comparison of academic digital 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	No	No	No	No
DQPHDE (Proposed Work)	Yes	Yes	Yes	Yes

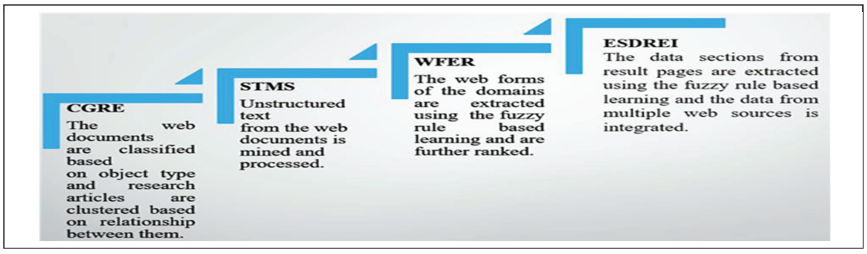


Figure 1 Phases of the proposed work.

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 Figure 1.

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). All the four modules

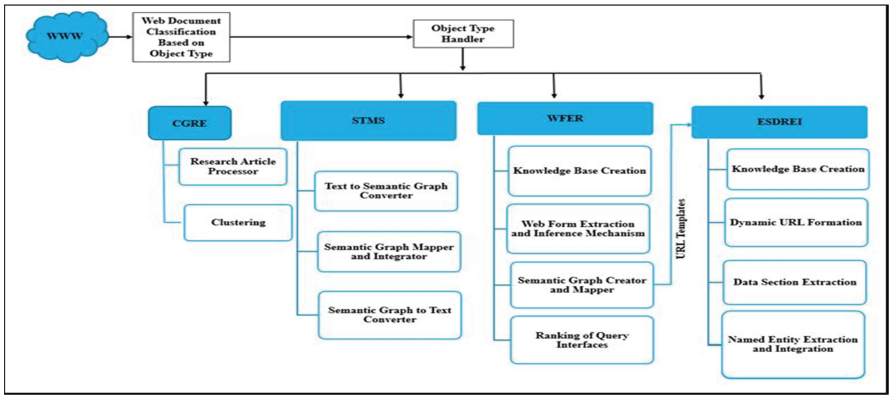


Figure 2 Parallel execution of the proposed system modules.

classified work in parallel. After the classification of downloaded documents, these are sent to theirrespectiveobject handlers and the processing is carried out independently by each module in parallel as shown in Figure 2.

3.1 Clustering Using Graph and Relationship Extraction (CGRE)

The detailed study of data lying on web 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 clusters the research articles using the graph and the relationship strength between the web articles. CGRE 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.

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" as shown in Figure 3.

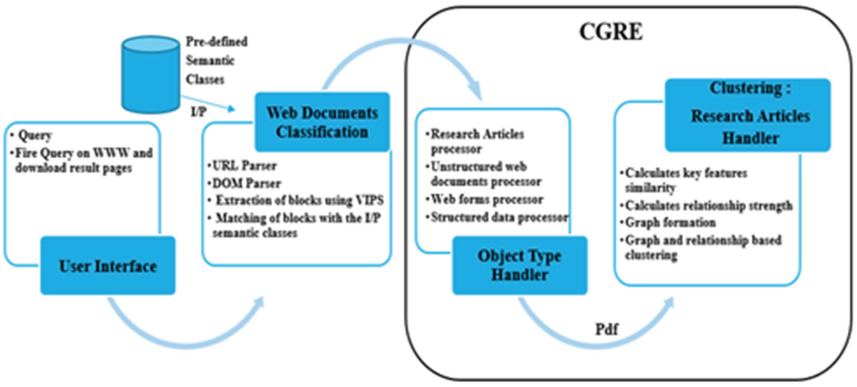


Figure 3 Architecture of CGRE.

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 contain the research articles that are clustered using the proposed technique CGRE, which uses the weighted relationship graph.

3.1.1 Web Document Classification

The downloaded web pages and semantic classes created acts as input to web document classifier. 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 [20]. 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 Figure 2. The classified web documents are sent to their respective object type handler.

3.1.2 Object Type Handler

The different handlers process the classified web documents in parallel and independently as shown in Figure 5

The research article processor works on pdf files. It clusters the articles based on significant key features, creates weighted graph of files and clusters

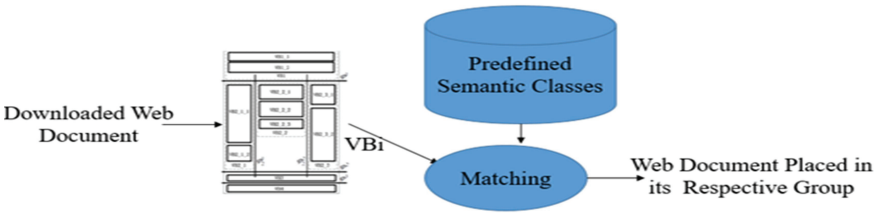


Figure 4 Web document classification.

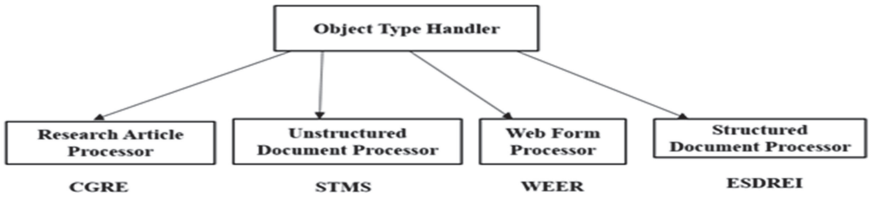


Figure 5 Object type handler.

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 ranks them. Structured document processor uses the rules and facts to extract the data region from the resultant pages.

3.1.3 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 [21]. 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. There are three steps in Research Article Processor namely, Key Feature Extraction, Key-Feature Similarity Calculation and Weighted Graph Creation.

3.1.4 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

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+; 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 6 Graph and relationship based clustering algorithm.

the web documents in the graph using proposed algorithm (CGRE) as shown in Figure 6.

3.2 Semantic Based Text Mining and Summarization (STMS)

Semantic based Text Mining and Summarization (STMS) proposes an algorithm. Semantic Graph Mapper and Integrator (SGMI) integrate and summarize 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

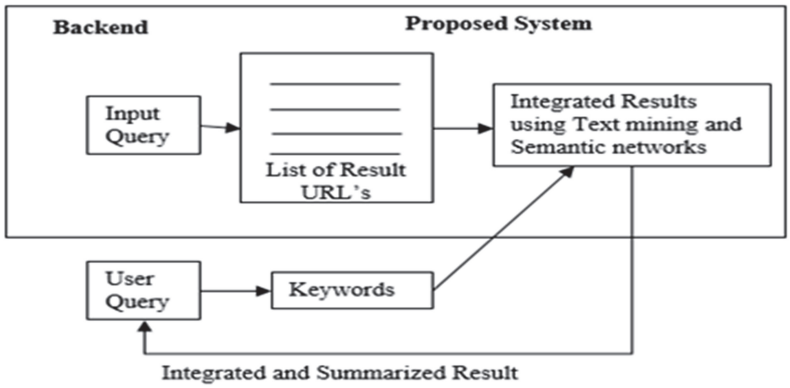


Figure 7 Proposed STMS architecture.

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 Figure 7.

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 therein 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 Figure 8.

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

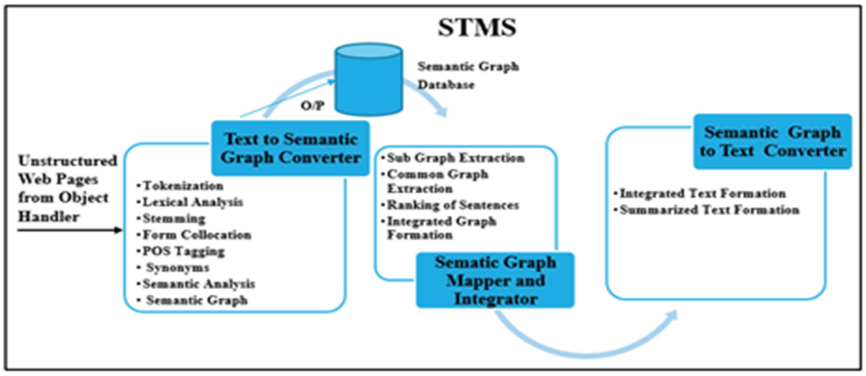


Figure 8 Architecture of STMS.

will issue a query, the integrated and the summarized result are shown to the user.

3.2.1 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 Figure 9.

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

- Tokenization
- Stemming
- Synonym Finding
- Form Collocation
- Tagging
- Semantic Analysis

In semantic graph creation the tagged tokens and the relationships between them are used as the input and the semantic graph is generated as

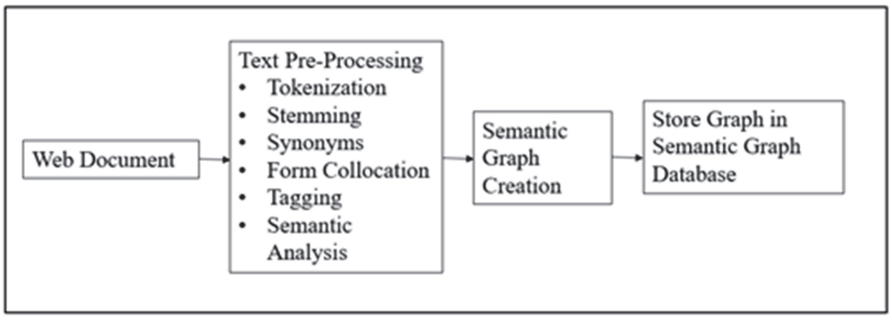


Figure 9 Text to semantic graph converter.

***Algorithm**: Text to Semantic Graph Converter*
***Input**: Web document Set (WD1……WDn)*
***Output**: Semantic Graph Database*

Algorithm:

For(i=1 to n) // n is total number of web documents in the set
{ For each sentence in web document
{
{ 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
}
{Create a semantic graph of tagged tokens,
store the location ofthe sentence and assign occurrence frequency =1 to the current processed sentence
}
{Link the sentence to the existing graph of the current web document.}
{Store the semantic graph of the web document in the semantic Graph Database. }

Figure 10 Text to semantic graph converter algorithm.

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 Figure 10.

3.2.2 Semantic Graph Mapper and Integrator

This module takes semantic graph database as input that results into summarized and integrated graph as depicted in Figure 11.

Input: *Semantic Graph Database*

Output: *Integrated and Summarized Semantic Graph*

Algorithm:

Integrated Graph (IG)= First Graph from the SGD
For i=2 to n (Total number of networks in the SGD)
{ Map the synonyms in IG and SGD [1] }
{ Sub-Graph = difference between the IG and SGD [1] }
{ Common graph= IG intersection SGD [i] }
{ Increment the frequency of sentences in the common graph 2.5. IG= Merge the sub-graph and common graph}
Extract the frequently occurring sentences from the IG and create the summary

Figure 11 Semantic graph mapper and integrator.

Figure 12 Semantic graph mapper and integrator algorithm.

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. The algorithm of SGMI is shown in Figure 12.

3.2.3 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.

3.3 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

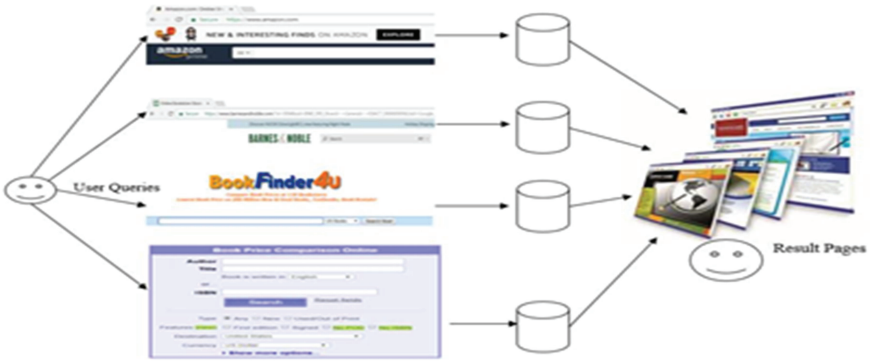


Figure 13 Interaction of end users with the web forms.

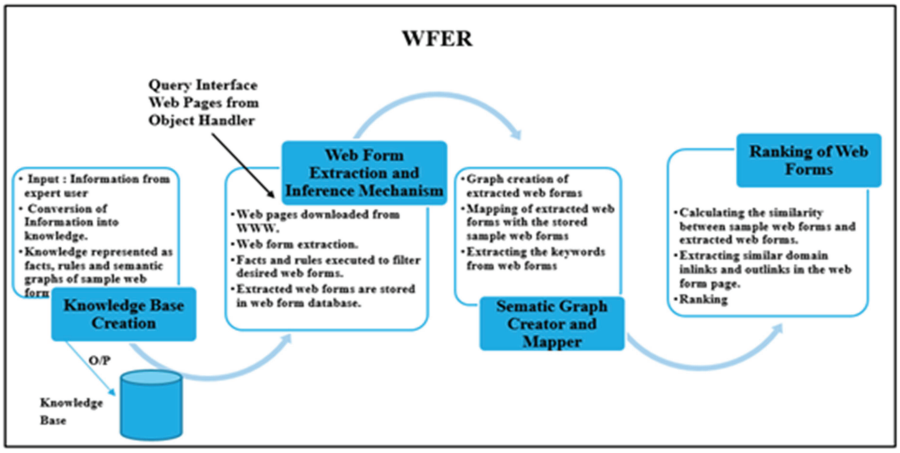


Figure 14 Modules of web form extraction and ranking of web forms.

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 Figure 13.

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 sub-modules of WFER are shown in Figure 14.

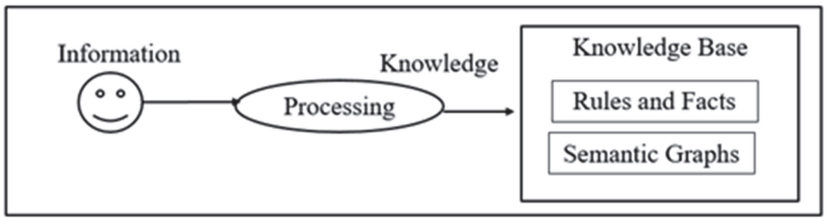


Figure 15 Knowledge base creation.

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 replaced 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.

3.3.1 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 Figure 15.

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. 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

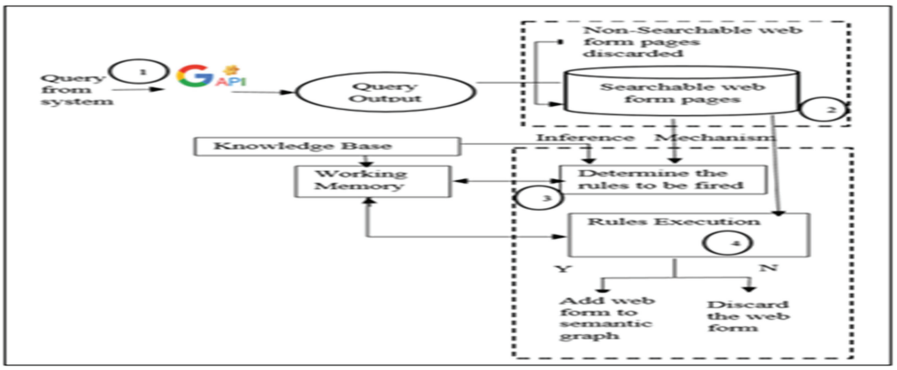


Figure 16 Web form extraction and inference mechanism.

(SSBWF) and multiple search box web forms (MSBWF). The basic structure of both these web forms now gets stored as semantic web form graphs in the knowledge base. 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.

3.3.2 Web Form Extraction and Inference Mechanism

The input to this module is the knowledge base and the downloaded web pages from the WWW.Theoutputof the module is 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 Figure 16.

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 Figure 17.

3.3.3 Semantic Graph Creator and Mapper

The extracted and processed web forms from previous module are now fed as input to this module. The output of the module is the mapped semantic graphs

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 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 17 Web form extraction and inference mechanism.

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. 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.

3.3.4 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. The number of out-links in a URL also get stored in the URL node as a tuple (URL, total number of out-links (TOL)) and it also contributes to ranking of web forms. For example, bookboon.com has a link to subsites.bookboon.com. More out-links result in raising the rank. The

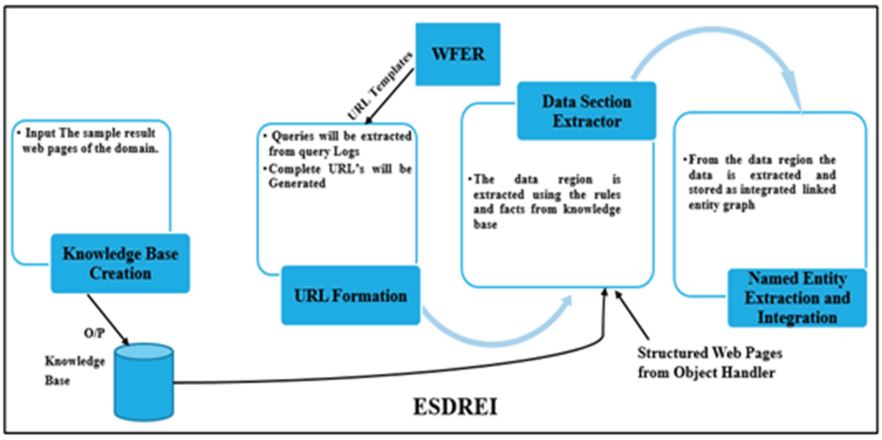


Figure 18 Proposed architecture of ESDREI.

third component in ranking the web form is the distribution of the benefit count of the in-link or the parent node in the web graph among its child nodes. There are three steps for ranking the web form namely keyword weight calculation and similarity finder, benefit calculation using the out-links, and rank calculation.

3.4 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 resulted 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 Figure 18. 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.

There are four steps in the proposed technique ESDREI namely knowledge base creation, URL formation, data section extractor and named entity extraction and integrator.

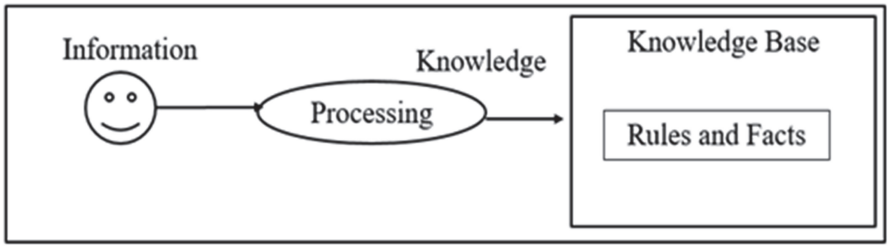


Figure 19 Knowledge base creation.

3.4.1 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 Figure 19.

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. The high-quality information about the sections of 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 of inference mechanism.

3.4.2 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 template 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". 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. These web pages will act as an input to the Data Section Extractor.

3.4.3 Data Section Extractor

The input to data section extractor module is the result web pages and the knowledge base which contains the rules as shown in Figure 20.

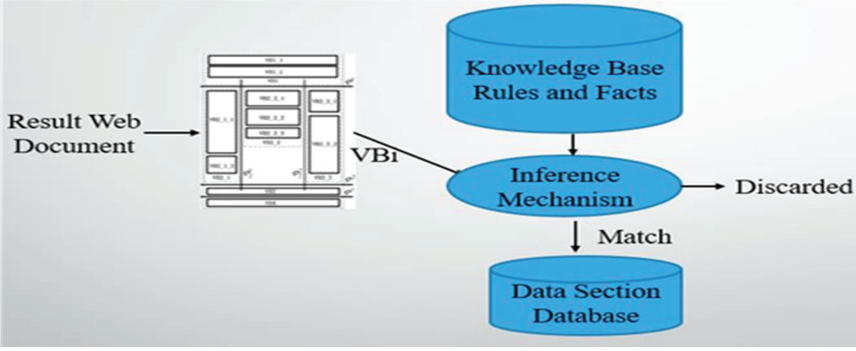


Figure 20 Data section extractor.

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.

3.4.4 Named Entity Extractor and Integrator

The inputs 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 store 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 the above 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.

4 Results Discussion

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 8 GB RAM using windows 10 using .NET C# as front-end and SQL 2000 as backend.

4.1 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 Figure 21. 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.

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 Figure 22.

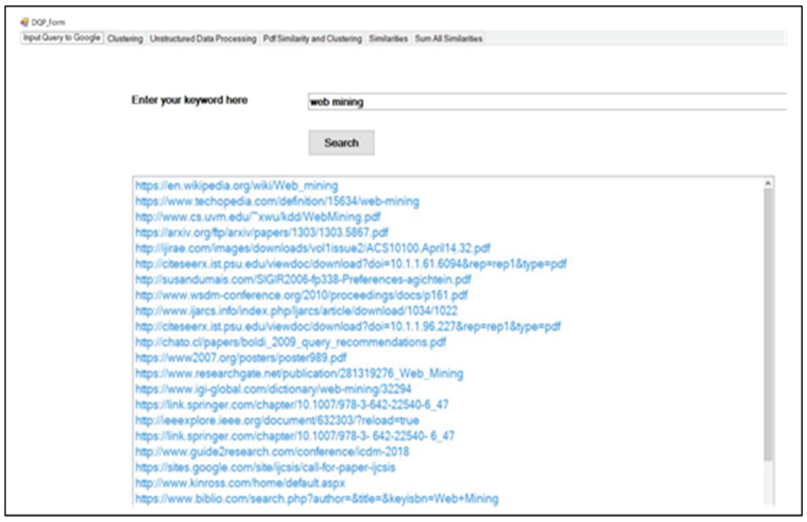


Figure 21 Home page for the system administrator.

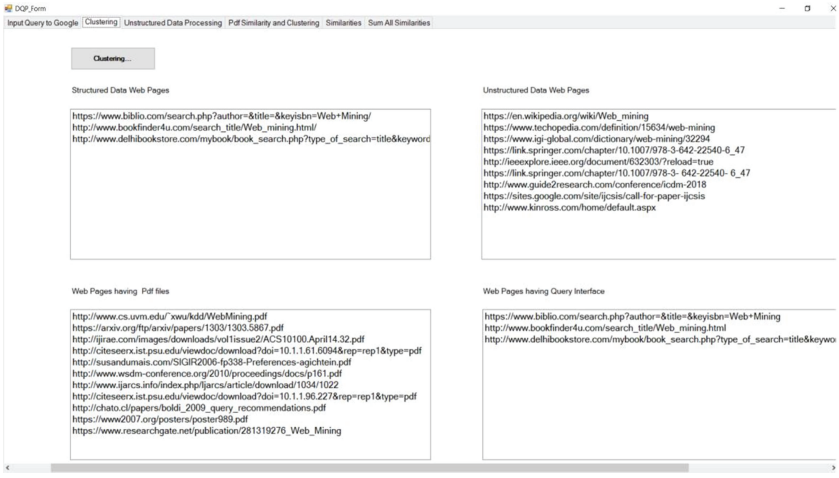


Figure 22 Classification of downloaded web documents.

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

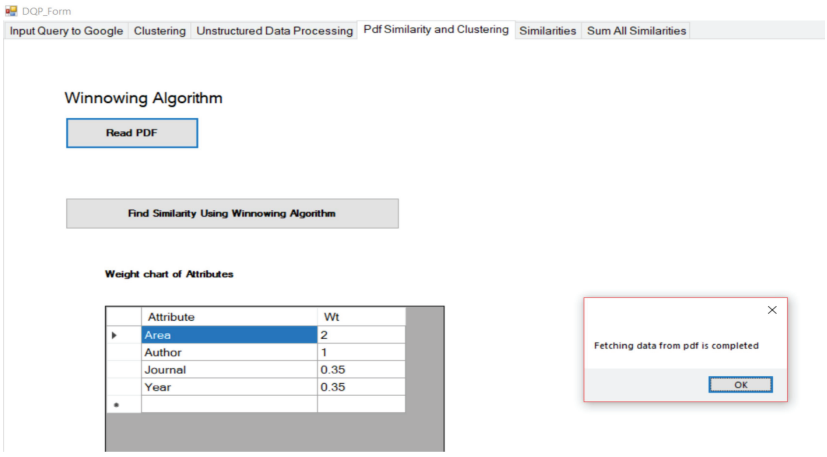


Figure 23 Key feature extraction from research articles.

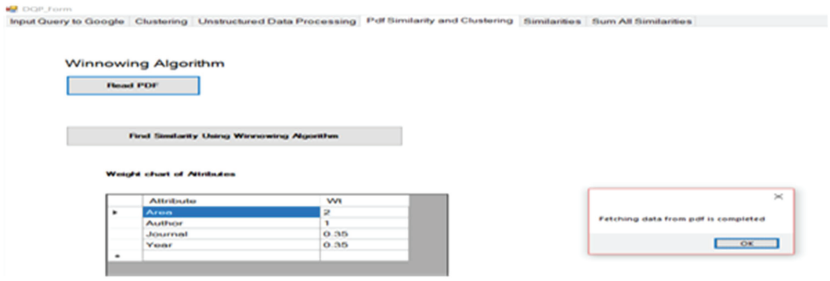


Figure 24 Similarity calculation of extracted key features.

processed and the values of these key features are fetched as shown in Figure 23.

Then the similarity of the key features is calculated using the winnowing algorithm as shown in Figure 24.

The similarity of the key features using the Winnowing Algorithm is extracted and the results of similarity are shown Figure 25.

It is followed by the calculation of relationship strength between the research articles by the help of the proposed work as shown in Figure 26.

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 Figure 27.

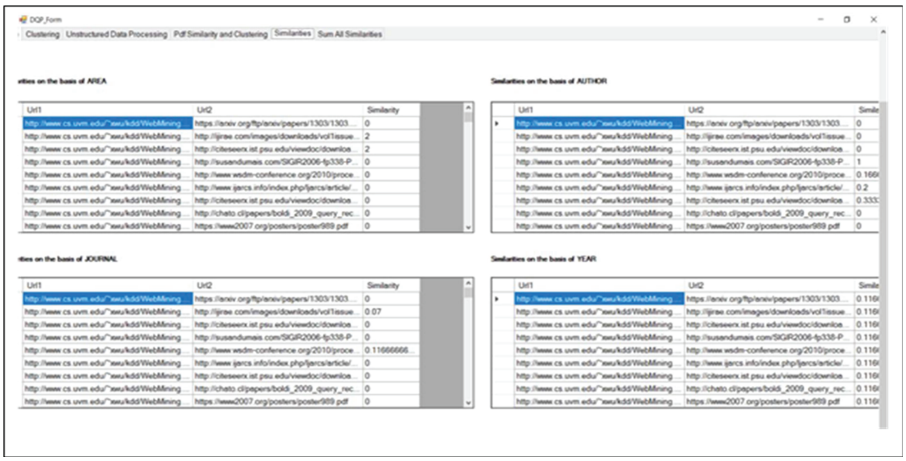


Figure 25 Results of similarity calculation of key features.

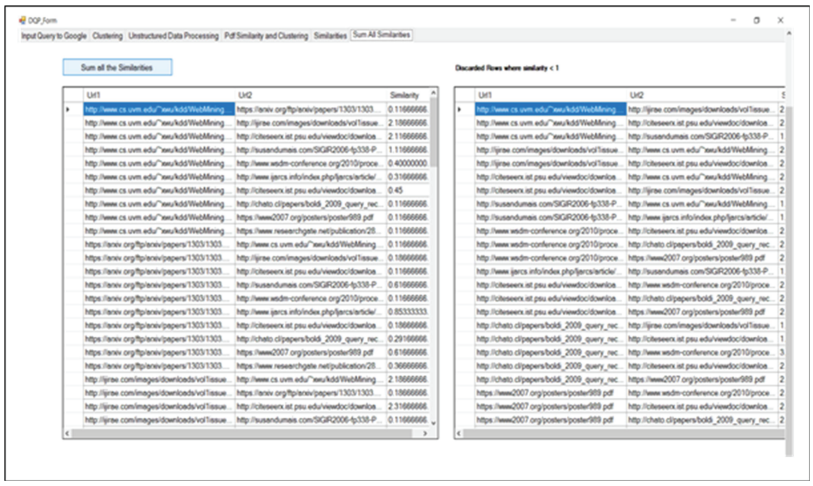


Figure 26 Calculation of relationship strength between research articles.

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 Figure 28. The irrelevant web pages are now removed from the document set.

The unstructured web pages that contain the tutorials are now processed by STMS module. The web documents are now get stored in the form of

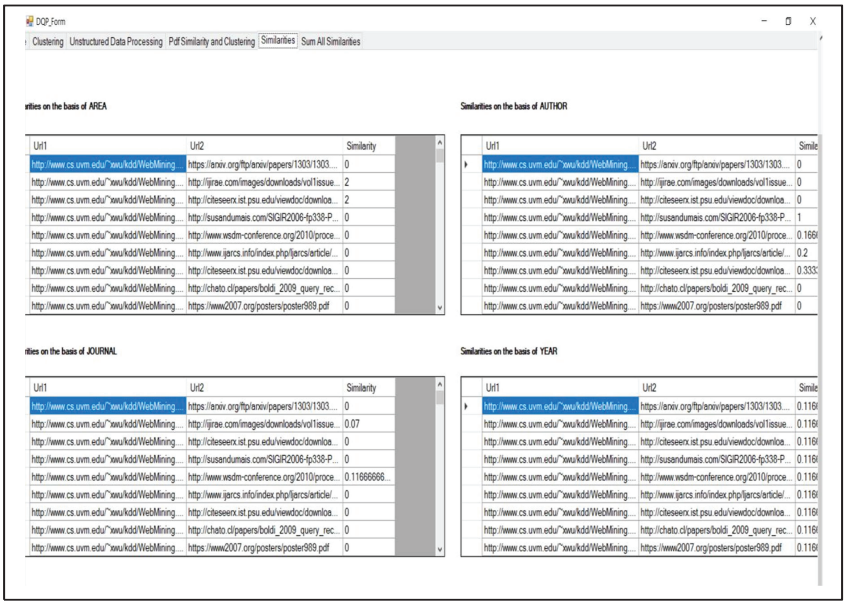


Figure 27 Clustering of research articles based on strength of relationship.

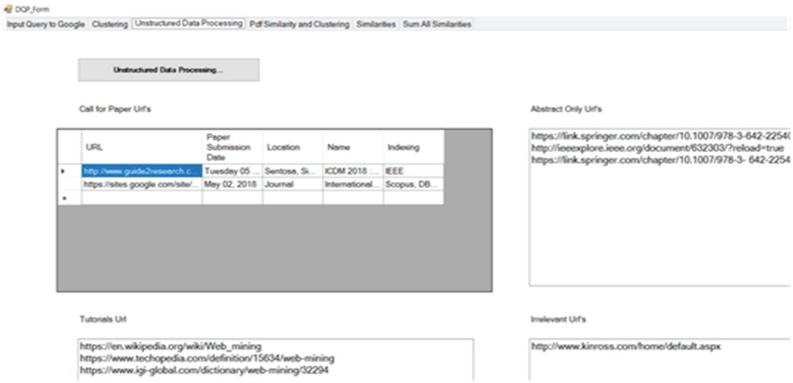


Figure 28 Categorization of unstructured web pages.

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 Figure 29.

It is followed by finding the unique content from the stored graphs of different web documents of a particular area as shown in Figure 30.

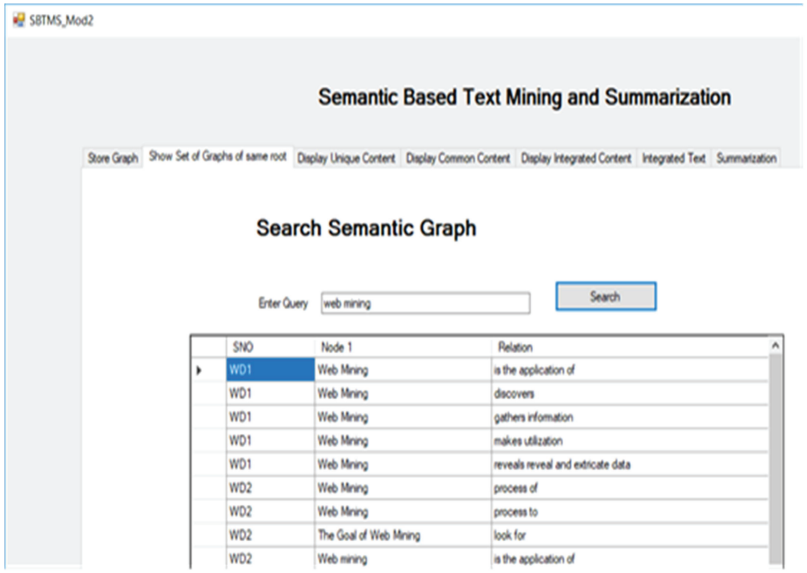


Figure 29 Sample of web document stored as graph.

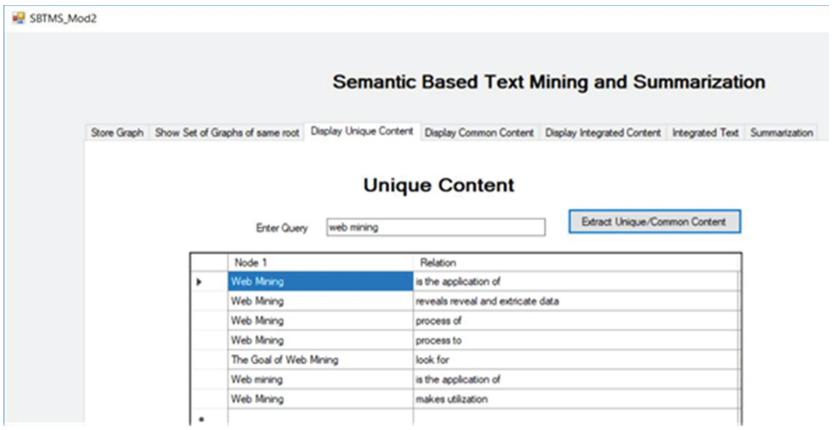


Figure 30 Extracting unique sentences from the graph of word documents.

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 Figure 31.

It is followed by the integration of data from multiple web pages. The duplicate sentences are already removed when common content is extracted.

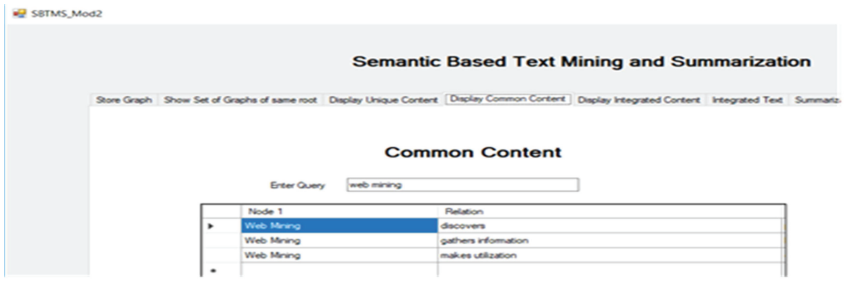


Figure 31 Extracting common sentences from the graph of word documents.

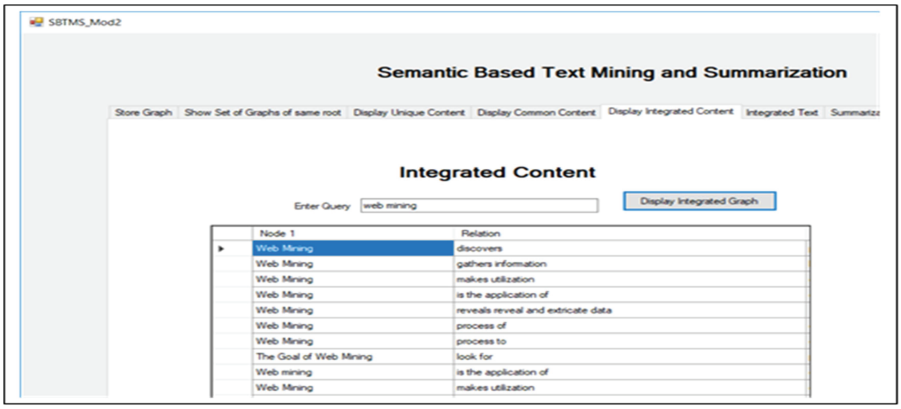


Figure 32 Integrated content from web documents in the form of graph.

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 Figure 32.

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 Figure 33.

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 Figure 34.

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 Figure 35.

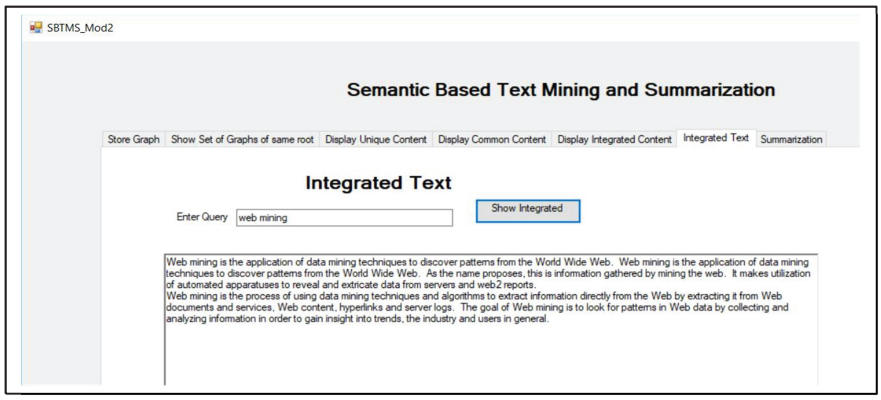


Figure 33 Integration of data from multiple web pages.

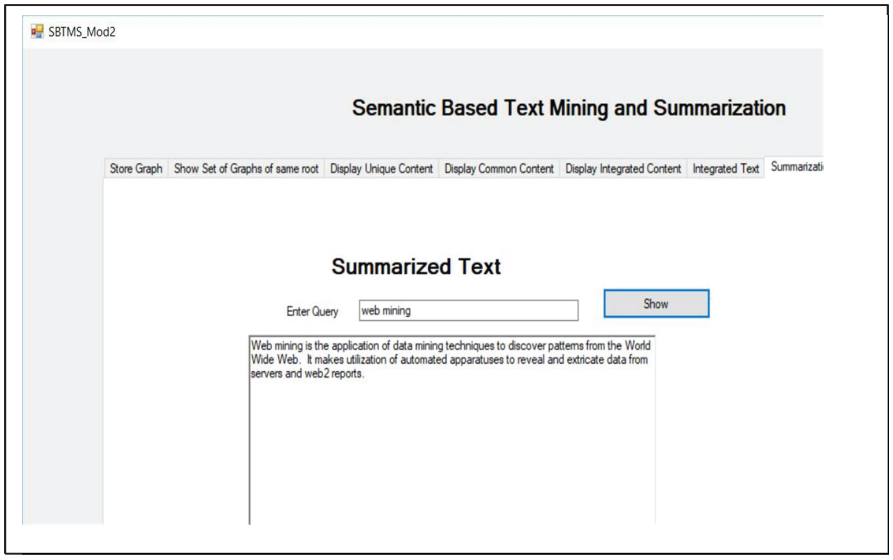


Figure 34 Summary of text from multiple web pages.

Next, the extracted web forms are ranked based on the in-links out-links and the benefit of the web page which contains that web form as shown in Figure 36.

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

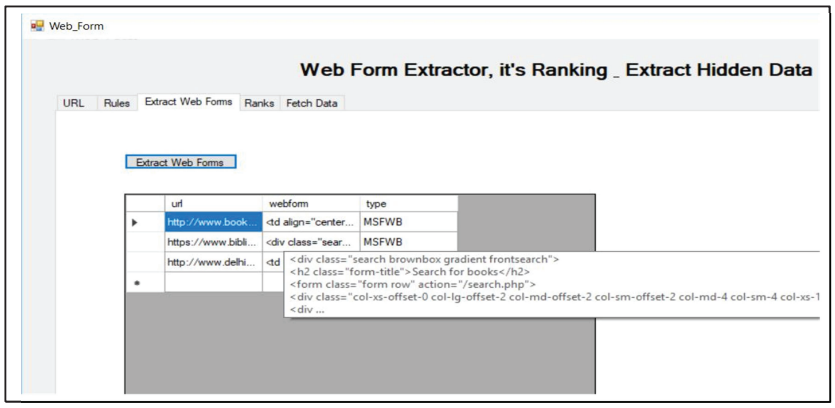


Figure 35 Extraction of web forms from web pages.

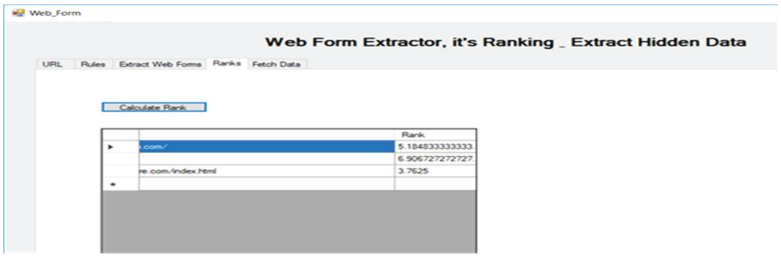


Figure 36 Ranking of web forms.

Figure 37 Processed and integrated data from the structured web pages.

provided by object handler is now processed by ESDREI and the data is extracted as shown in Figure 37.

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

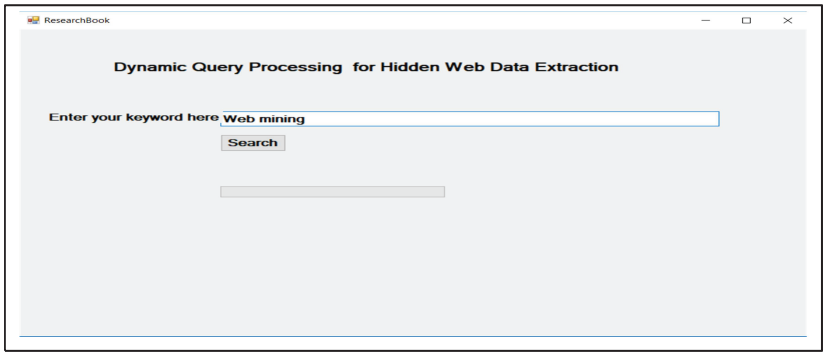


Figure 38 Home page of end user.

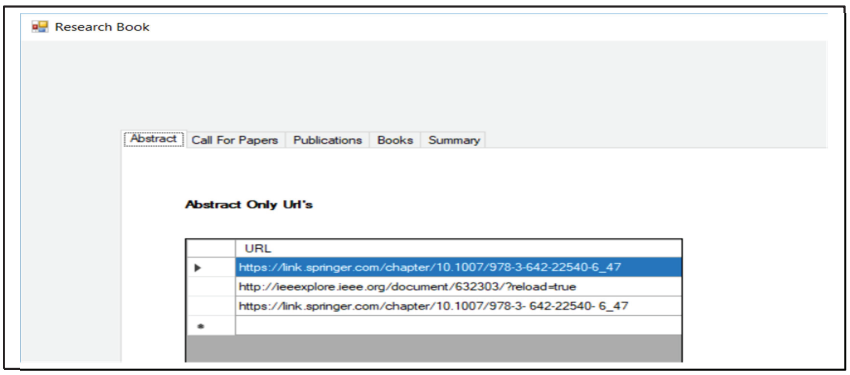


Figure 39 The links of web pages containing the abstracts.

End User Side

When the end user logins into the system home page, it is shown in Figure 38. 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.

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 Figure 39. The first tab of the result page shows the URL's where only the abstract of the research papers are stored.

The research articles are clustered using CGRE module of the proposed work and the results are shown in Figure 41. The user can group the research

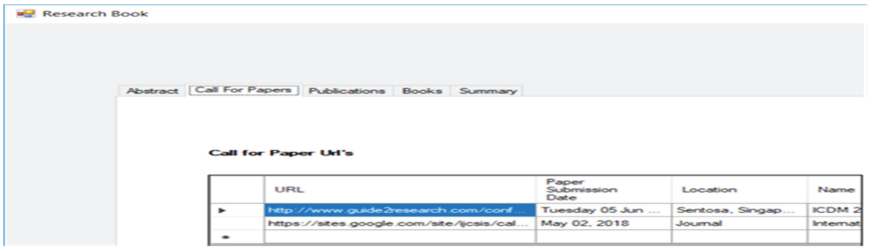


Figure 40 The call for papers data and links of those URL's.

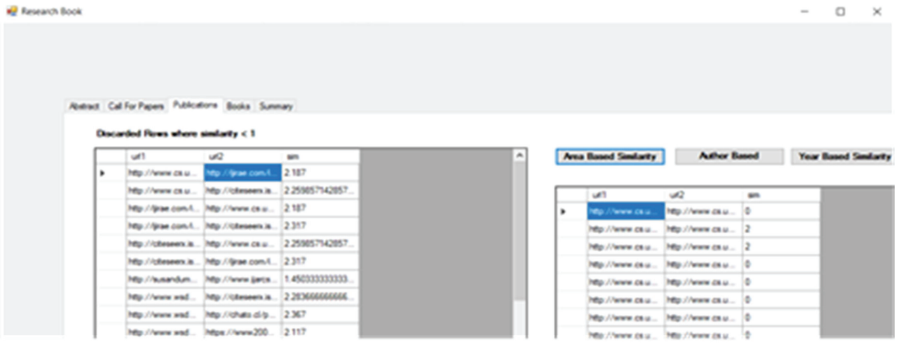


Figure 41 Clustering of research articles.

articles at run time also by clicking on his/her required key feature of clustering.

In the fourth tab the data from the hidden web is extracted by WFER and ESDREI module of the proposed work as shown in Figure 42. ESDREI extracts and integrates the data from multiple web pages. It removes the redundancy and combines the details of the entity which is scattered across various pages in WWW.

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 Figure 43.

4.2 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 is not only interested in the research articles, they

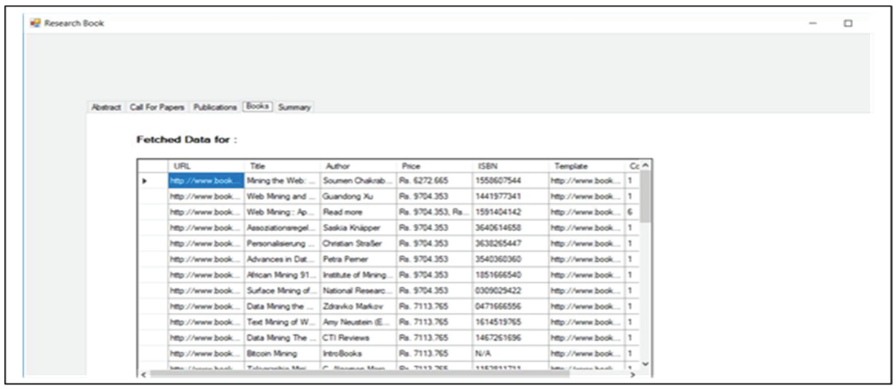


Figure 42 Data extracted from structured web pages.

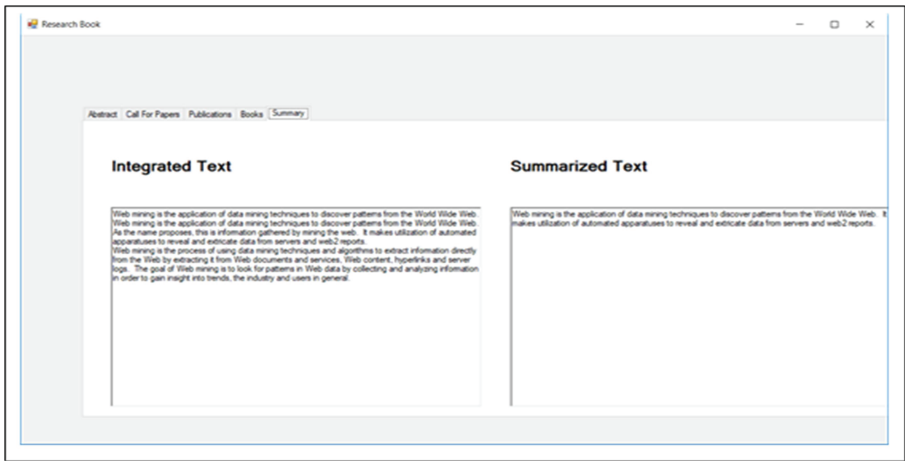


Figure 43 The integration and summary of unstructured web pages.

can also look for certain books relevant to their area. Then they can also go through the research articles which are 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

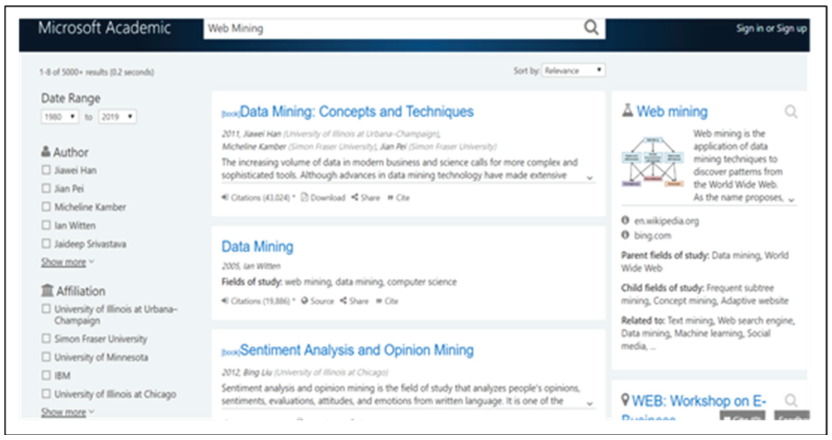


Figure 44 Result page of microsoft academic.

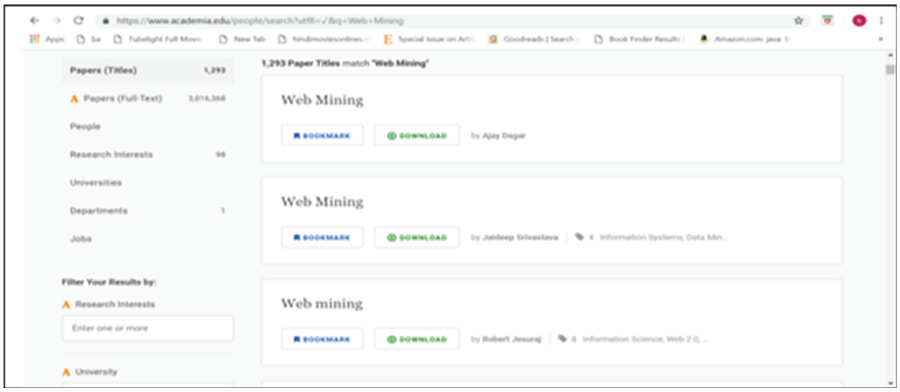


Figure 45 Result page of Academia.edu.

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 information from surface web and hidden web information, a single portal is developed from where the user can get all the data required.

Table 2 Comparison between 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

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 Figure 46 the user has fired the query "Data Mining" and the result page returned by Microsoft Academic is shown in Figure 44. 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 Figure 45 and it shows the research articles only. The user is not allowed to filter the research articles according to his/her requirements at run time. The other kind of information needed by the researcher is also not available.

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 2.

5 Conclusion

DQPHDE proposed to extract and process the information from surface web as well as hidden web. 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 all portions of WWW. 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.

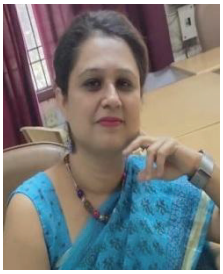
References

[1] Kunder, Maurice de. World Wide Web Size. 2011.
[2] https://www.internetworldstats.com/stats.html.
[3] http://www.internetlivestats.com/one-second/.
[4] Eduard c. Dragut, Weiyi Meng and Clement T. Yu (2012). Deep web query interface understanding and integration. Synthesis Lectures on Data Management. https://doi.org/10.2200/S00419ED1V01Y201205DTM026.
[5] Michael k. Bergman (2001).The deep web: surfacing hidden value Bright Planet Corp.
[6] B. He, M. Patel, Z. Zhang, and K. C.-C. Chang (2007). Accessing the deep web. Communications of the ACM, 50(5). pp. 94–101.
[7] https://www.popsci.com/dark-web-revealed.
[8] https://doaj.org/about.
[9] https://core.ac.uk/about.
[10] Pieper, Dirk (May 18, 2011). BASE Migration InetBib, www.base-search.net.
[11] https://en.wikipedia.org/wiki/Google_Scholar.
[12] Microsoft. Academic Knowledge API Retrieved 29 January 2017.
[13] ResearhGate.net.
[14] Academai.edu.
[15] B. He and K. C.-C. Chang (2003) Statistical Schema Matching across Web Query Interfaces. In SIGMOD. doi:10.1145/872757.872784.
[16] A. D. Sarma, X. Dong, and A. Halevy (2008). Bootstrapping pay-as-you-go data integration systems In SIGMOD. doi:10.1145/1376616.1376702.
[17] Disheng Qiu, Luciano Barbosa, Xin Luna Dong (2015). DEXTER: Large-Scale Discovery and Extraction of Product Specifications on the Web. Proceedings of the VLDB Endowment. doi:10.14778/2831360.2831372.
[18] Manuel Álvarez, Juan Raposo, Fidel Cacheda and Alberto Pan (2006).A Task specific Approach for Crawling the Deep Web. Engineering Letters, EL_13_2_19 (Advance online publication).
[19] https://en.wikipedia.org/wiki/Academia.edu.
[20] Deng Cai, Shipeng Yu,Ji-Rong Wen and Wei-Ying Ma (2003). VIPS: A Vision based Page Segmentation Algorithm. Technical Report MSR-TR-2003-79, Microsoft Technical Report.

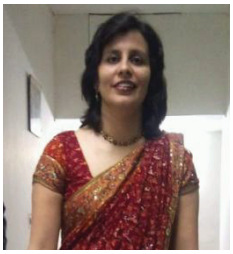
[21] SchleimerSaul, Wilkerson Daniel S., Aiken Alex (2003) Winnowing: Local Algorithms forDocument Fingerprinting. ACM SIGMOD international conference on Management of data, NY, USA, 2003, pp. 76–85.

Biographies

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.



Anuradha Pillai is an Associate Professor in the Department of Computer Engineering, JC Bose University of Science and Technology, YMCA, Faridabad, Haryana, India. She received Ph.D. in Computer Engineering from Maharishi Dayanand University, Rohtak. She published more than 60 papers in reputed international journals and successfully guided 4 PhD students. Her subjects of interest include Data Mining, Information Retrieval, Hidden web, Web Mining and Social Networks.



Deepika Punj is working as Assistant Professor in Department of Computer Engineering at JC BOSE University of Science and Technology YMCA, Faridabad, India. She has done Ph.D in Computer Engineering. She is having 14 years of experience in teaching. She has published more than 25 papers in Reputed National and International Journals. Her research interests include Data Mining, Deep Learning, Machine Learning and Internet Technologies.

Jyoti Verma, received her PhD degree in the year 2011. She has a teaching experience of 17 years and research experience of 9 years. She is credited with 35 research articles to her name in the journals of repute. Her research interests include Information Retrieval, Web Mining and Big Data. She is currently working as Associate Professor in the Department of Computer Engineering, J.C. Bose University of Science and Technology, Faridabad, India.