

Web page content block partitioning for Focussed Crawling

*Thesis submitted in partial fulfillment of the requirements for the award of
degree of*

**Master of Engineering
in
Software Engineering**

Submitted By
Aastha
(Roll No. 801031001)

Under the supervision of:

Dr. Deepak Garg

Associate Professor



**COMPUTER SCIENCE AND ENGINEERING DEPARTMENT
THAPAR UNIVERSITY
PATIALA – 147004
June 2012**

CERTIFICATE

I hereby certify that the work which is being presented in the thesis entitled, “**Web page content block partitioning for Focussed Crawling**”, in partial fulfillment of the requirements for the award of degree of Master of Engineering in *Software Engineering* submitted in Computer Science and Engineering Department of Thapar University, Patiala, is an authentic record of my own work carried out under the supervision of *Dr. Deepak Garg* and refers other researcher's work which are duly listed in the reference section.

The matter presented in the thesis has not been submitted for award of any other degree of this or any other University.

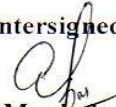

Aastha

This is to certify that the above statement made by the candidate is correct and true to the best of my knowledge.



Dr. Deepak Garg
Associate Professor
CSE Department
Thapar University
Patiala

Countersigned by


(Dr. Maninder Singh)

Head
20/1/12
Computer Science and Engineering Department
Thapar University
Patiala


(Dr. S. K. Mohapatra)
Dean (Academic Affairs)
Thapar University
Patiala

Acknowledgment

I express my sincere and deep gratitude to my guide Dr. Deepak Garg , Associate Professor in Computer Science & Engineering Department, Thapar University Patiala, for the invaluable guidance, support and encouragement. He provided me all resource and guidance throughout thesis work.

I am heartfelt thankful to Dr. Maninder Singh, Head of Computer Science & Engineering department Thapar University Patiala, for providing us adequate environment, facility for carrying out thesis work.

I would like to thank to all staff members who were always there at the need of hour and provided with all the help and facilities, which I required for the completion of my thesis. At last but not the least I would like to thank God and mine parents for not letting me down at the time of crisis and showing me the silver lining in the dark clouds.

Aastha
801031001

Abstract

The World Wide Web (WWW) is a collection of billions of documents formatted using HTML. Web Search engines are used to find the desired information on the World Wide Web. Whenever a user query is inputted, searching is performed through that database. The size of repository of search engine is not enough to accommodate every page available on the web. So it is desired that only the most relevant pages must be stored in the database. So, to store those most relevant pages from the World Wide Web, a better approach has to be followed. The software that traverses web for getting the relevant pages is called “*Crawlers*” or “*Spiders*”.

A specialized crawler called focused crawler traverses the web and selects the relevant pages to a defined topic rather than to explore all the regions of the web page. The crawler does not collect all the web pages, but retrieves only the relevant pages out of all. So the major problem is how to retrieve the relevant and quality web pages.

To address this problem, in this thesis, we have designed an algorithm which partitions the web pages on the basis of headings into blocks and then calculates the relevancy of each partitioned block in web page. Then the page relevancy is calculated by sum of all block relevancy scores in one page. It also calculates the URL score and identifies whether the URL is relevant to a topic or not. As compared to previous methods of partitioning, our method on the basis of headings is more appropriate because in other methods, sub tables of a table are considered to be the other block. But it is not so. These must be the part of that block only in which the table resides. On the basis of headings, there is an appropriate division of pages into blocks because a complete block comprises of the heading, content, images, links, tables and sub tables of a particular block only.

Table of Contents

Certificate	ii
Acknowledgment	iii
Abstract	iv
Table of Contents	v
List of Figures	viii
Chapter 1: Introduction	1-10
1.1 Overview	1
1.2 Working of Web	1
1.3 Search Engine	2
1.4 Technique of search engines	2
1.5 Web Crawlers	2
1.6 Types of search engines	3
1.6.1 Crawler based search engines	4
1.6.2 Human powered search engines	6
1.6.3 Hybrid search engines	7
1.7 Definition of web crawlers	7
1.8 Definition of Focussed Crawling	7
Chapter 2: Literature Survey	8-21
2.1 A survey of web crawlers	8
2.2 Basic Crawling Terminology	11
2.3 Parallel Crawlers	12
2.4 Crawling Techniques	13
2.4.1 Distributed Crawling	13
2.4.2 Focussed Crawling	13
2.5 System architecture of Focussed crawler	14

2.5.1 Seed pages fetching subsystem	15
2.5.2 Topic keywords generating subsystem	16
2.5.3 Similarity Computing engine	17
2.6 Pseudo code of a basic web crawler	18
2.7 Algorithms used in focussed crawling	20
2.8 Algorithm of focussed crawler	20
Chapter 3: Problem Statement	22-23
Chapter 4: Proposed Algorithm and Implementation	24-35
4.1 Definition of Segmentation	24
4.2 The Proposed Approach	24
4.3 Content Block Partitioning	25
4.4 Algorithm of Block Partitioning	25
4.5 Focussed Crawling procedure	29
4.5.1 Topic specific weight table construction	29
4.5.2 Block Analysis	29
4.5.3 URL score Calculation	29
4.5.4 Algorithm of URL score Calculation	30
4.6 Dealing with Irrelevant pages	31
4.6.1 Pseudo code of algorithm	31
4.7 Results and Discussions	33
4.7.1 Performance Metrics	34
Chapter 5: Conclusions and Future Work	36-37
5.1 Conclusion	36
5.2 Future Work	37
References:	38-41

List of Figures

Figure No.	Figure Title	Page No.
Figure 1	Simplified Web crawler	3
Figure 2	Crawler Based search engines	5
Figure 3	Directories of search engines	6
Figure 4	Standard and Focussed crawling	8
Figure 5	Components of web crawler	12
Figure 6	Structure of parallel crawlers	13
Figure 7	Focussed crawler working process	15
Figure 7(a)	Fetch seed pages	16
Figure 7(b)	Build/update	17
Figure 7(c)	Similarity computing engine	18
Figure 8	A snippet of HTML pages	27
Figure 9	The tag tree of a block	27
Figure 10	Partitioning of web pages into blocks	28
Figure 11 (a)	Focused crawling process without tunneling	32
Figure 11(b)	Focussed crawling process with tunneling	32

List of Tables

Table No.	Table Title	Page No.
Table 1	Comparison of focussed and non focussed Algorithms.	10

1.1 Overview

The World Wide Web (or the Web) *is a collection of billions of interlinked documents formatted using HTML*. WWW is a network where we can get a large amount of information. In a Web, a user views the Web pages that contains text, images, and other multimedia and navigates between them using hyperlinks. The Internet and the World Wide Web are not same. The Internet is a collection of interlinked networks that are linked by wires, fiber-optic cables, wireless connections, etc. Whereas the Web is a collection of interconnected documents linked by hyperlinks and URLs. The World Wide Web is one of the services of the Internet, along with various others including e-mail, file sharing etc. However, "the Internet" and "the Web" can be used interchangeable non-technically. To publish the information on internet we need *search engines*. Because it is not possible to handle all this data by humans manually. So people used what they are looking for on WWW by using search engines like Google, Yahoo!

1.2 Working of Web [1]

To view a Web page on the World Wide Web, the procedure starts by typing the URL into a Web browser, or by following a hyperlink to that page. The Web browser then gives some messages in order to fetch and display it. First, the server-name of the URL is resolved into an IP address that uses the domain name system, or DNS. This IP address is used to send data packets to the Web server. The browser then requests the resource by sending an HTTP request to the Web server at that given address. In the case of a common Web page, the HTML text of the page is requested first and then parsed by the Web browser, which will then make requests for images and other files. All this searching within the Web is performed by the special engines that are known as Web Search Engines [2].

1.3 Search Engine

By Search Engine, we are usually referring to the actual search that we are performing through the databases of HTML documents .It is software that helps in locating the information stored on WWW [1].

1.4 Technique of how search engine presents information to the user initiating a search

When you ask a search engine to get the desired information, it is actually searches through the index which it has created and does not actually searches through the Web. Different search engines give different ranking results because not every search engine uses the same algorithm to search through all the indices.

1.5 The question is what is going on behind these search engines and why is it possible to get relevant data so fast?

The answer is web crawlers. The web crawler is a software program that traverses the web by downloading the pages and follows the links from page to page. Such programs are also called wanderers, robots, spiders, and worms. The structure of the World Wide Web is a graphical structure, *i.e.* the links of a page are used to open other web pages. Internet is a directed graph, web page as node and hyperlink as edge, so the search operation is a process of traversing the directed graph. By following the linked structure of the Web, we can traverse a number of web-pages starting from a seed page. Web crawlers are used to create a copy of all the visited pages for later processing by a search engine that will index the downloaded pages that will help in fast searches. Web search engines work by storing information about many web pages, which they retrieve from the WWW. These pages are retrieved by a Web crawler. Web *crawlers* are programs that use the graph structure of the web to move from page to page.

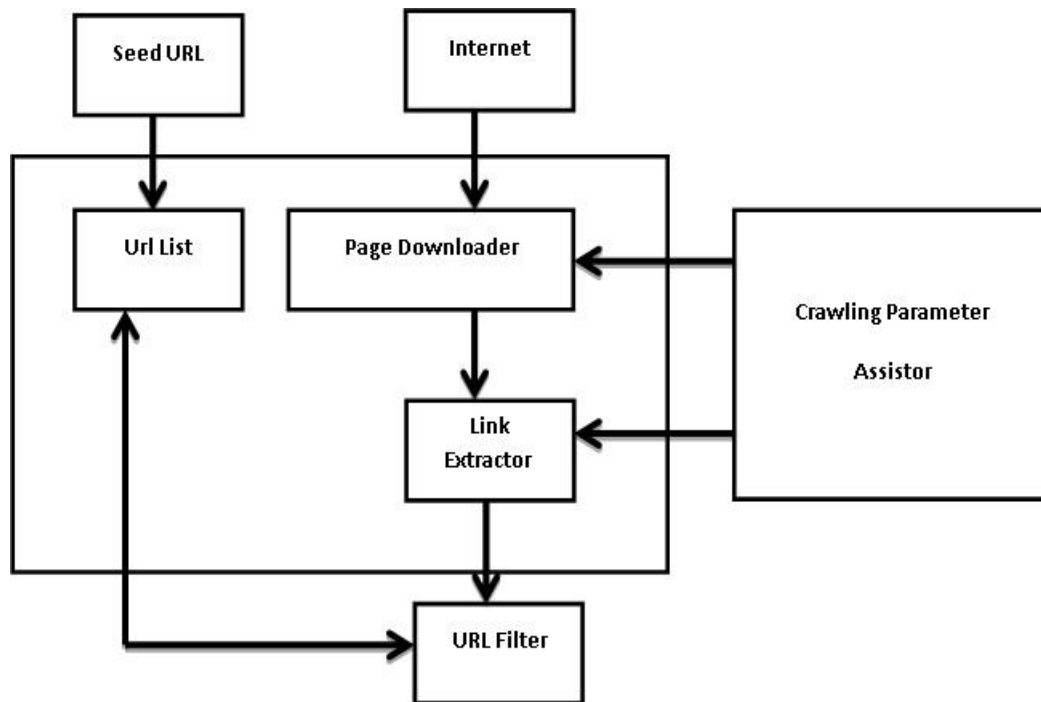


Figure 1: A simplified web crawler [3]

It is a simplified Web crawler in Figure 1. According to Figure 1, a Web crawler starts from a URL called the Seed URL to visit the Internet. The Page Downloader gets a URL from URL List to download the page and gives page to the Link Extractor. The Page Downloader checks whether to download pages or not. As the crawler visits these URLs, the Link Extractor identifies all the hyperlinks whether they are according to the requirements and transfers them to the URL Filter, and finally stores the results into URL list. The Crawling Parameter Assistor provides the parameter setting for the needs of all parts of the crawler.

Web crawler was internet's first search engine that has performed keyword searches in both names and texts of the page. It was developed by Brain Pinkerton, a computer student at the University of Washington [4].

1.6 Types of Search Engines:

The search engine belongs to 3 different categories and all are unique. All are having different rules and procedures. There are basically 3 types of search engines [2, 5]

- Those that are powered by robots (called *crawlers, ants or spiders*)

- Those that are powered by human submissions
- Hybrid search engines.

1.6.1 Crawler Based Search Engine:

Such search engines use crawlers to categorize the web pages. Crawlers visit a Web Site to find information on the internet and store it for search results in their databases. A crawler finds a Web page, downloads it and analyzes the information presented on the web page. The web page will then be added to the search engine's database. When a user performs a search, the search engine will check its database of Web pages for the keywords the user searched. The results are listed on the pages by order of which is closest. Although they usually aren't visible to someone using a Web Browser.

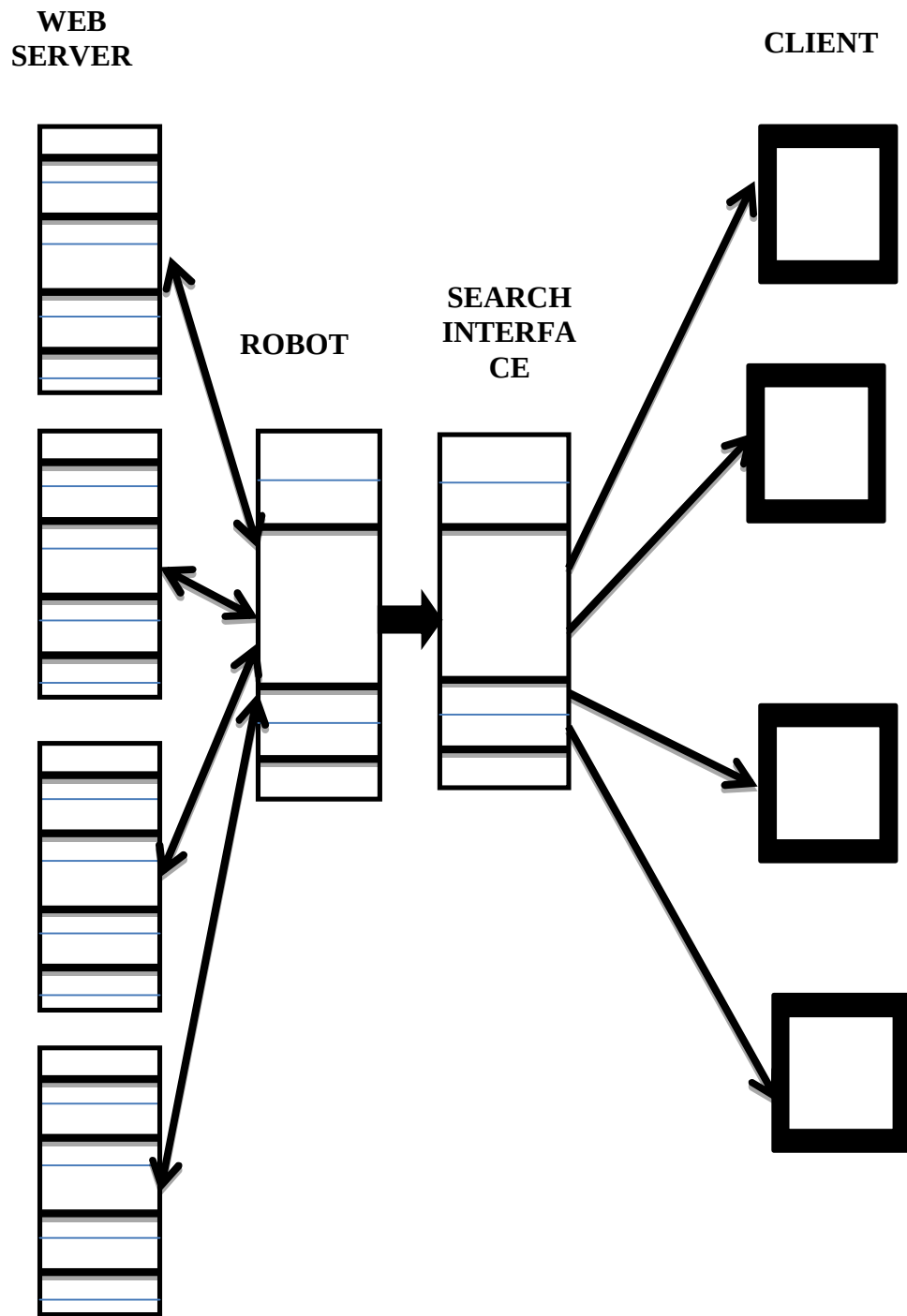


Figure 2: A crawler based search engine [6]

1.6.2 Human-powered Search Engines

Such search engines rely on humans to give information that is indexed. Only information that is submitted by humans is indexed. This type of search engines are

mostly used at small scale and rarely used at large scale. A Directory uses human editors that decide the site belongs to which category. They place Websites in 'directories' database. By focusing on particular categories, user narrows the search to those records that can be relevant. The human editors occasionally check the website and rank it, based on the information they find using some set of rules.

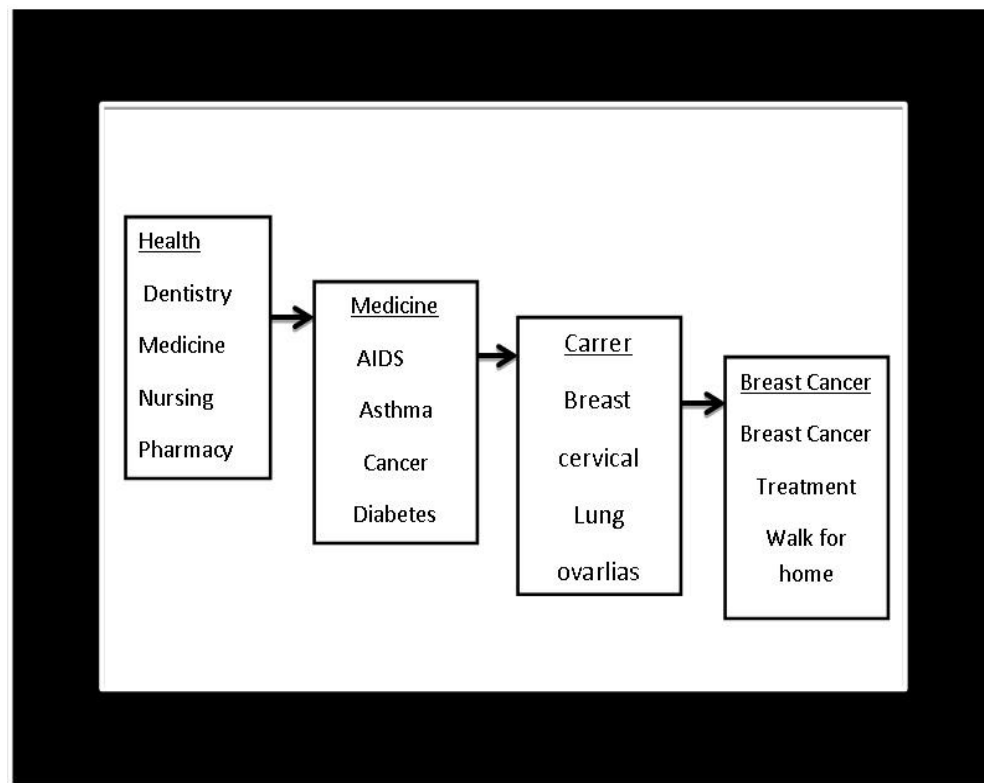


Figure 3: Directories of a search engine [6].

Looksmart, Lycos, AltaVista, MSN, Excite and AOL search relied on providers of directory data to make their search results more meaningful.

1.6.3 Hybrid Search Engines

Hybrid search engines use a combination of both crawler based results and directory results. It differs from traditional search engine such as Google or a directory based search engine such as yahoo in which the programs operates by comparing a set of metadata. Examples of hybrid search engines are: Yahoo, Google.

1.7 Definition of Web-Crawler

A web-crawler is a program or automated script which browses the World Wide Web in a methodical and automated manner. To move from page to page web crawlers use the graphical structure of the Web [2, 7]. Such programs are also called wanderers, robots, spiders, and worms. The World Wide Web has a graphical structure, *i.e.* the other pages are opened by traversing the links given in a page. Actually Internet is a directed graph, web page as node and hyperlink as edge, so traversing the directed graph is the search operation. Web *crawlers* are programs that exploit the graph structure of the web to move from page to page. However 'crawlers' itself doesn't indicate the speed of these programs, so they are known as fast working programs [8].

1.8 Definition of Focussed Crawling

The information can be used to collect more on related data by intelligently and efficiently choosing what links to follow and what pages to discard. This process is called *Focused Crawling* [9]. Focused crawling is a promising approach for improving the precision and recall of search on the Web. It is a crawler that will seek, acquire, index, and maintain pages on a specific topic. Such a focused crawler entails a very small investment in hardware and network resources and achieves desired results.

Chapter 2

Literature Survey

2.1 A Survey of Web Crawlers [10]

The original Google crawler [2,11] was developed at Stanford). Topical crawling was first introduced by Menczer. Focused crawling was first introduced by Chakrabarti et al. [10,12] A focused crawler has the following components: (a) How to know whether a particular web page is relevant to given topic, and (b) way to determine how to follow the single page to retrieve multiple set of pages. A search engine which used the focused crawling strategy was proposed in [18] based on the assumption that relevant pages must contains only the relevant links. So it searches deeper where it finds relevant pages, and stops searching at pages not as relevant to the topic. But, the above crawlers are having a drawback that when the pages about a topic are not directly connected the crawling can stop at early stage. They keep the overall number of downloaded Web pages for processing [13] to a minimum while maximizing the percentage of relevant pages. For high performance, the seed page must be highly relevant. Seed pages can also be selected among the best results retrieved by the Web search engine [14, 15].

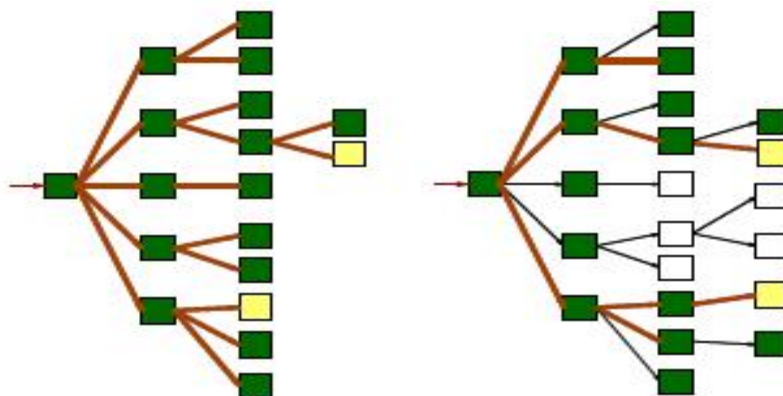


Figure 4: (a) Standard Crawling

(b) Focussed Crawling

A standard crawler followed a breadth first strategy. If the crawler starts from a web page which is n steps from a target document, we have to download before all the documents that are up to $n-1$ steps from the starting document.

b) A focused crawler identifies the most relevant links, and ignores the unwanted documents. If the crawler has to start from document that is n steps from target document, it downloads a subset of the documents that are maximum $n-1$ steps from the starting document. If the search strategy is optimal, then the crawler takes only n steps to discover the target.

A focused crawler efficiently seeks out documents about a specific topic and guides the search based on both the content and link structure of the web [9]. Figure 4 graphically illustrates the difference between a breadth first crawler and a typical focused crawler. A focused crawler implements a strategy that associates a score with each link in the pages it has downloaded. [16, 17, 18].

A topical crawler ideally downloads only web pages that are relevant to a particular topic and avoid downloading the irrelevant pages. So a topical crawler can predict the probability that a link to that page is relevant before actually downloading the page. A predictor can be the anchor text of links; and this approach was taken by Pinkerton [19]. Menczer et al. [20] show that simple strategies are very effective for short crawling, while techniques such as reinforcement learning [21] and evolutionary adaptation gives the best performance for longer crawling. Diligenti et al. [22] use the complete content of the pages that are visited already to get the similarity between the query and the pages that have not been visited yet. Guan et al [23] propose a new frontier prioritizing algorithm which efficiently combines link-based and content based analysis to evaluate the priority of an uncrawled URL in a queue.

Approaches to focused crawling are Best first approach, Infospiders, Fish search and Shark search. In Best first approach, [24] we have given a Frontier of links and the next link is selected on the basis of some priority or score. So every time a best available link is opened and traversed. Infospiders uses neural networks. Info Spiders [25, 26] is a multi-agent system for online, dynamic Web search. Fish search [27] is based on the assumption that relevant pages must have relevant neighbors. Thus, it searches deeper on the documents that are found relevant to the search query, and do not search in "dry"

areas. In Fish-search algorithm Internet is treated as a directed graph, webpage as node and hyperlink as edge, so the search operation is the process of traversing directed graph. For every node we judge whether it is relevant, 1 means the node is relevant and 0 for irrelevant. So all the relevant pages are assigned the same priority value. The list of URLs which is maintained are having different priority, the URL which are at the front of the list are more superior, and will be searched sooner than others. If relative page is found, it stands for that the food has been found by the fish. However Fish Search algorithm has some limitations, so a powerful improved version of Fish Search algorithm is developed known as- *Shark Search*. [28]

In this algorithm, the improvement is that instead of the binary (relevant/irrelevant) evaluation, it returns a "fuzzy" score, i.e., a score between 0 and 1 (0 for no similarity and 1 for perfect "conceptual" match) rather than a binary value. In shark search we have found a threshold value which can determine the relevance of the page. However, Best first crawlers have been shown better results in case of infospiders and shark search and other non focussed breadth first crawling approaches. So, best first crawling is considered to be the most successful approach to focused crawling due to its simplicity and efficiency.

Table 1: Comparison of focussed and non focussed algorithms

Non focussed Algorithm	Focussed Algorithms	
Breadth first search:	Approaches	
It uses the frontier as a FIFO queue, crawling links in the order in which they are encountered. The problem with this algorithm is that when the	1. Best First search	From a given Frontier of links, next link for crawling is selected on the basis of some priority or score. Thus every time the best available link is opened and traversed.

frontier is full, the crawler can add only one link from a crawled page. since it does not use any knowledge about the topic, it acts blindly. That is why, also called, Blind Search Algorithm.	2.Fish Search	For every node we judge whether it is relevant, 1 for relevant, 0 for irrelevant. Therefore all relevant pages are assigned the same priority value.
	3.Shark Search	Rather than using binary (relevant/irrelevant) evaluation of document relevance, it returns a "fuzzy" score, i.e., a score between 0 and 1 (0 for no similarity and 1 for perfect "conceptual" match)

2.2 Working of Basic Web Crawler

The structure of a basic crawler is shown in figure 5 [2, 29].

The basic working of a web-crawler can be discussed as follows:

1. Select a starting seed URL or URLs.
2. Add it to the frontier.
3. Now pick the URL from the frontier.
4. Fetch the web-page corresponding to that URL.
5. Parse that web-page to find new URL links.
6. Add all the newly found URLs into the frontier.
7. Go to **step 2** and repeat while the frontier is not empty.

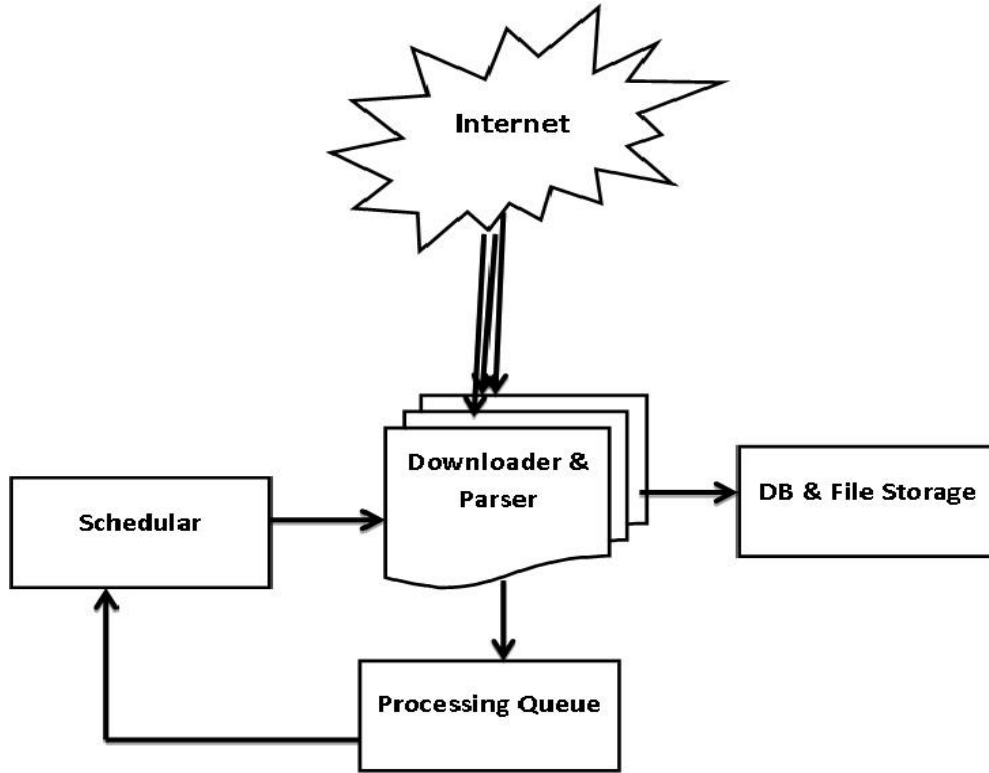


Figure 5: Components of a web-crawler [29]

Note that it also depicts the 7 steps given earlier. Such crawlers are called sequential crawlers because they follow a sequential approach.

2.3 Parallel Crawlers

The size of the Web grows exponentially, so it is very difficult to retrieve the significant pages of the Web from a large number of web pages by using a single sequential crawler. Therefore, multiple processes are run by the search engines in parallel to perform the task of getting relevant pages, in order to maximize the download rate. We call this type of crawler as a *parallel crawler*. Parallel crawlers as the name indicates work parallelly to get the pages from the Web and add them to the database of the search engine [30].

The parallel crawling architecture is shown in the figure 6. Each parallel crawler have its own database of collected pages and own queue of un-visited URLs. Once the

crawling procedure finishes, the collected pages of every crawler are added to the database of the search engine. Parallel crawling architecture no doubt increases the efficiency of any search engine.

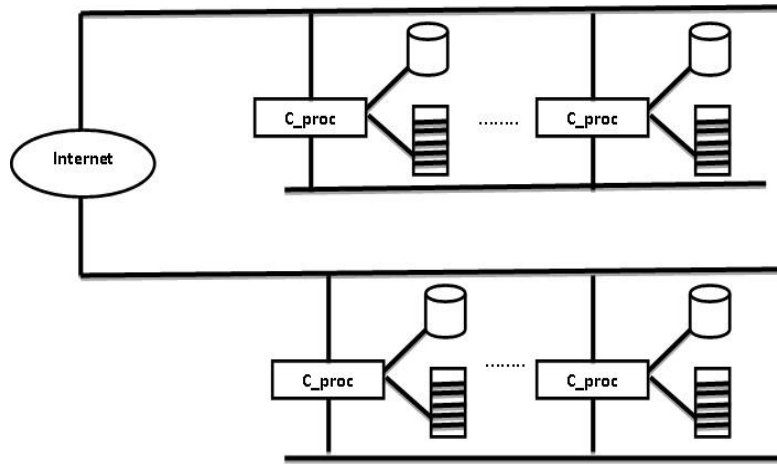


Figure 6: Structure of a Parallel Crawler [30]

2.4 Crawling Techniques [4]

2.4.1. Distributed Crawling

The size of web is A single crawler process even if it is a multithreading process will be insufficient for large search engines that have to fetch large amount of data in a very less time. When a single crawler is used all the fetched data passes through a single physical link. By distributing the crawling makes the system scalable and easily configurable and also makes the system fault tolerable.

2.4.2. Focussed Crawling

The goal of a focused crawler is to seek out pages that are selective and are relevant to a desired topic. Therefore a focused crawler can predict the probability that a link to a particular page is relevant before actually downloading the page [20]. The performance of a focused crawler depends on the richness of links in the specific topic being searched. The topics are specified not using keywords, but using the documents,

focused crawlers try to “predict” whether or not a target URL is pointing to a relevant web page before actually fetching the page. In addition, focused crawlers visit URLs in an optimal order such that URLs pointing to relevant and high-quality Web pages are visited first, and URLs that point to low-quality or irrelevant pages are never visited. This leads to significant savings in hardware and network resources, and helps to keep the crawl more up-to-date.

2.5 System Architecture of focused crawler [4]

The focussed crawler is made up of four subsystems:

1. Seed pages fetching subsystem.
2. Topic keywords generating subsystems.
3. Similarity computing engine.
4. A spider

The whole working process of the focused crawler is showed in figure 7.

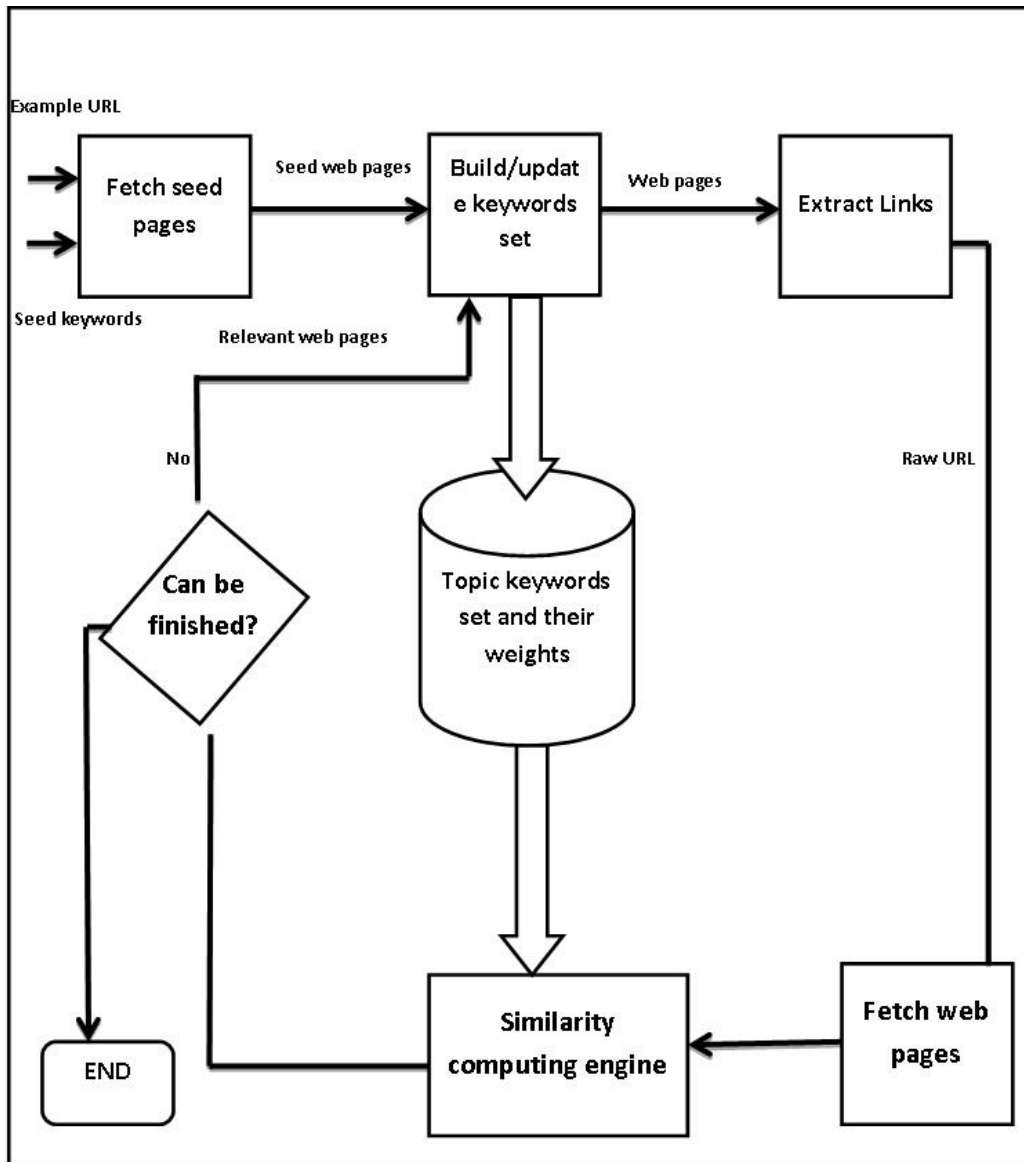


Figure 7: Focused Crawler working process [31]

2.5.1. Seed pages fetching subsystem

From the given seed keywords, the system searches them on a search engine. The result which is returned by search engine consists of a huge set. The top N ($N < 500$) URL's are probably relevant to the topic. The crawler uses these top N URLs as seed URLs and from these URLs, it fetches the seed pages. Fig.10 (a) shows how focused crawler generates seed pages.

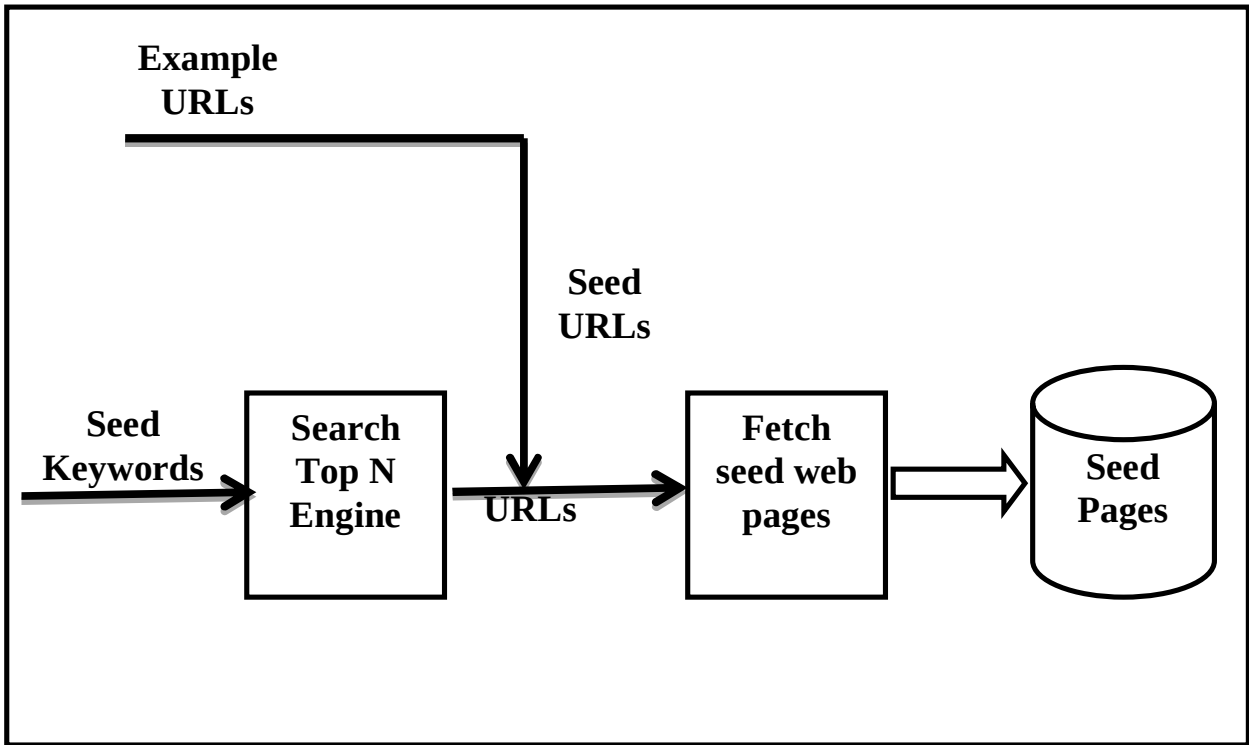


Figure 7(a): Fetch seed pages by seed keywords and example URLs [31]

2.5.2. Topic keywords generating subsystem

If the documents are mostly relevant to the topic, then it is easier to find the topic keywords from them and this subsystem is designed to find topic keywords from those documents. For each word T_i in document, first the term frequency tf is counted by the system, and then retrieve its document frequency df and finally computes weight. Weight (i). The top N ($N < 50$) highest weight keywords are outputted as topic keywords set.

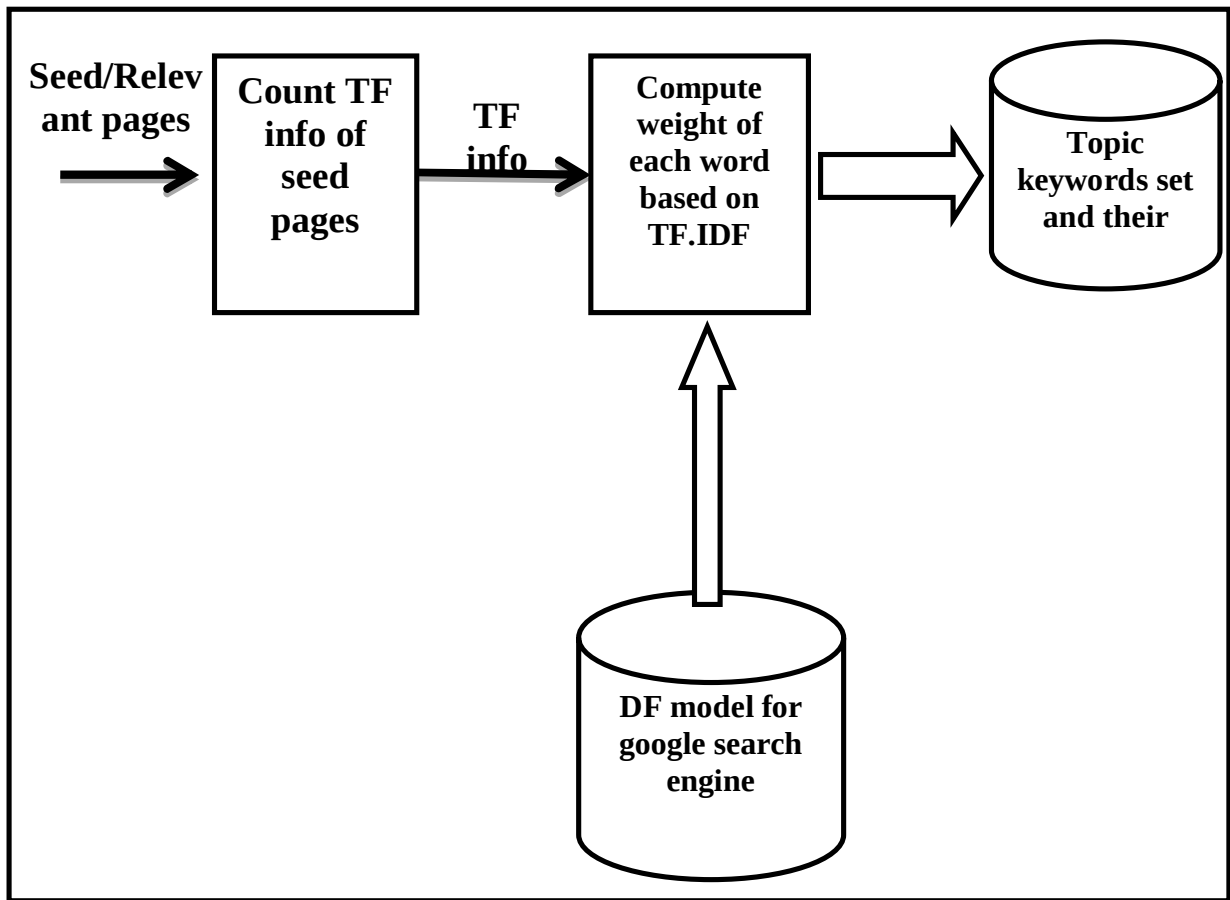


Figure 7(b): Build/update topic keywords by seed/relevant web pages [33]

2.5.3. Similarity Computing Engine

When a crawler fetches a new page, it needs to judge the page whether or not the page is relevant to the topic. The document D is that web page which has to be judged. The query Q is a set of topic keywords. The computing result is Similarity $\text{Sim}(Q, D)$ and its float value is between 0 and 1. We have set a threshold θ as a standard for judgement of document relevance. If the value of θ is higher, the precision of retrieved pages relevant to the topic would be higher. But the recall would be lower. Figure 10 (b) shows the procedure of similarity computing engine.

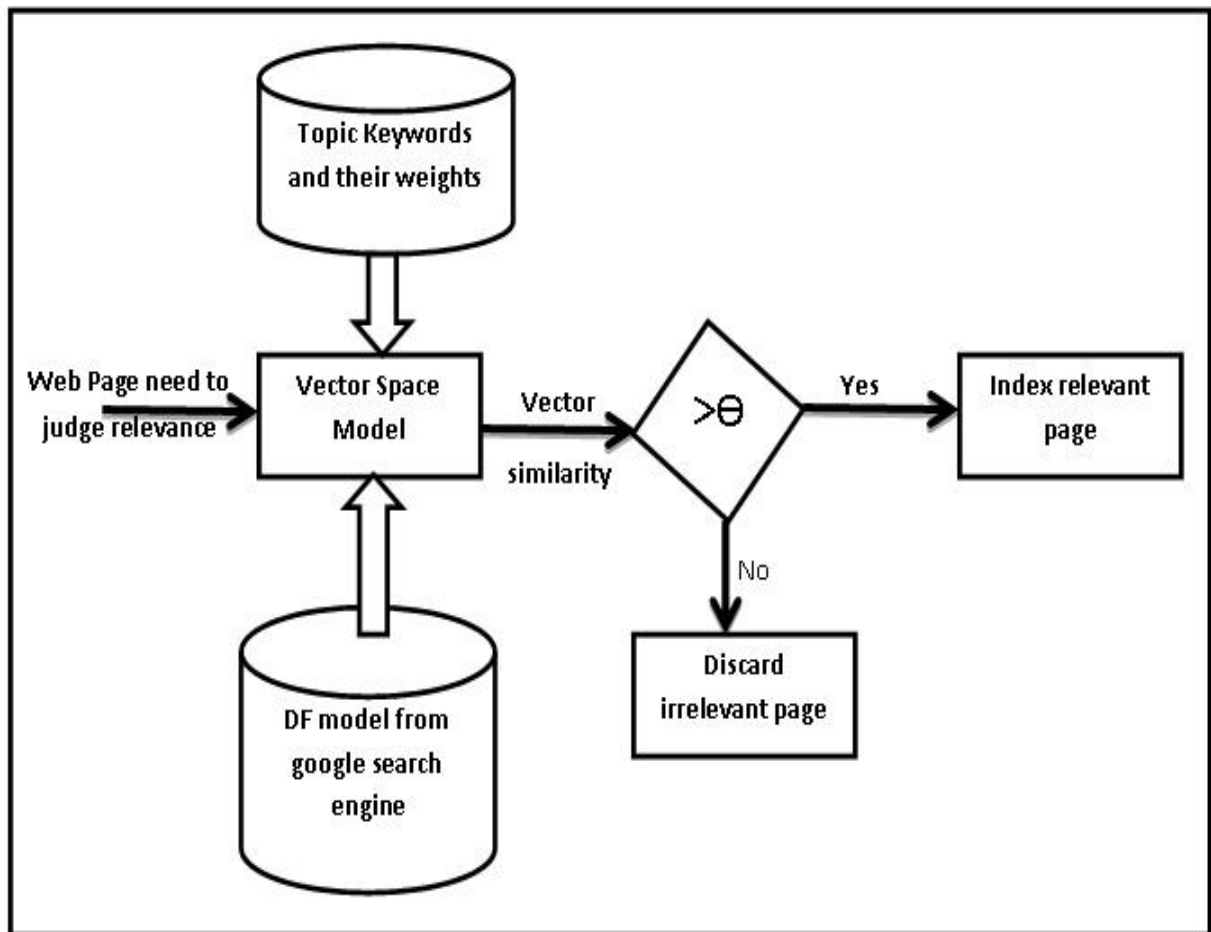


Figure 7(c): Similarity Computing Engine [31]

2.6 Pseudo code of a basic web crawler

Add the URL to the empty list of URLs to search

While not empty (the list of URLs to search)

{

Take the first URL in from the list of URLs

Mark this URL as already searched URL

If the URL protocol is not HTTP then

break ;

```
        go back to while

    If robots.txt file exists on site then
        If file includes Disallow statement then
            break ;
            go back to while

    Open the URL
        If the opened URL is not HTML file then
            break ;
            go back to while

    iterate the HTML file

    While the HTML text contains another link {

        If robots.txt file exist on URL/site then
            If file includes Disallow statement then
                break ;
                go back to while

        If the opened URL is HTML file then
            If the URL isn't marked as searched then
                Mark this URL as already searched URL

            Else if type of file is user requested
                Add to list of files found

    }
```

2.7 Types of Algorithms used in Focused Crawlers

Focused crawlers rely on two types of algorithms. *Web analysis algorithms* are used to estimate the relevance and quality of the Web pages and *Web search algorithms* determine the order in which the target URLs is visited.

2.8 An Algorithm of focussed crawler [32]

A focussed crawler algorithm which efficiently combines link based and content-based analysis to evaluate the priority of an uncrawled URL in the frontier.

Input: topic T, threshold of relevant of page content T1, threshold of relevant of text of linkage T2, threshold of count of crawling pages T3;

Output: Web pages relevant to topic

1. while (queue of linkage is not null)^(amount of crawling pages < T3) do
2. Get the linkage at the head of queue and downloading web page P the linkage linked and calculate the relevant topic T
Relevance (P) =similarity (P, T)
If relevance (P) <T1 then
3. Dismiss page P and all of linkages in this page;
4. goto 15:
5. end
6. for each linkages a in the page P do
7. Score a as follows:
 Relevance (a) =similarity (a, T)
 if relevance(a)<T2 then
 dismiss a;
9. goto 6;
10. end
11. if the linkage a has not been crawled then
12. add linkage a into queue of linkage

13. end

14. end

Chapter 3

Problem Statement

As the information on the WWW is growing so far, there is a great demand for developing efficient methods to retrieve the information available on WWW. Search engines present information to the user quickly using Web Crawlers. Crawling the Web quickly is an expensive and unrealistic goal as it requires enormous amounts of hardware and network resources. A focused crawler is software that aims at desired topic and visits and gathers only a relevant web page which is based upon some set of topics and does not waste time on irrelevant web pages. The focussed crawler does not collect all web pages, but selects and retrieves only the relevant pages and neglects those that are not concern. But we see, there are multiple URLs and topics on a single web page. So the complexity of web page increases and it negatively affects the performance of focussed crawling because the overall relevancy of web page decreases. A highly relevant region a web page may be obscured because of low overall relevance of that page. Apart from main content blocks, the pages have such blocks as navigation panels, copyright and privacy notices, unnecessary images, extraneous links, and advertisements. Segmenting the web pages into small units will improve the performance. A content block is supposed to have a rectangle shape. Page segmentation transforms the multi-topic web page into several single topic context blocks. This method is known as *content block partitioning*. In this thesis, we will present an algorithm how to efficiently divide the web page into content blocks and then we will apply focussed crawling on all the content blocks. A web page will be partitioned into blocks on the basis of *headings*. As compared to previous methods of partitioning, our method on the basis of headings is more appropriate because in other methods, sub tables of a table are considered to be the other block. But it is not so. These must be the part of that block only in which the table resides. On the basis of headings, there is an appropriate division of pages into blocks because a complete block comprises of the heading, content, images, links, tables and sub tables of a particular block only. First we make the HTML tag tree of a block. Each HTML page corresponds to a tree where tags are internal nodes and the detailed texts, images or hyperlinks are the leaf nodes.

When the pages are segmented into the content blocks, the relevant blocks are crawled further to extract the relevant links from them.

Then the relevancy value of each block is calculated separately and summed up to find the overall relevancy of the page. The relevancy of web page may be inappropriately calculated if the web page contains multiple topics that can be unrelated and that may be a negative factor. Instead of treating a whole web page as a unit of relevance calculation, we will evaluate each content block separately.

Proposed Algorithm and its Implementation

4.1 Definition of Segmentation

Focussed crawlers collect the pages on specific topic and ‘predict’ whether or not a target URL is pointing to a relevant page before actually fetching the page. The purpose of partitioning the web page into blocks is that first we partition the pages into blocks, then only those URLs are extracted which belongs to only the relevant blocks and do not extract those URLs which do not belong to relevant block. A problem faced by focused crawlers is that they measure the relevancy of a page and calculates the URL score of the whole page and a Web page usually contains both relevant as well as irrelevant topics. So, if we evaluate the whole page, lot of irrelevant links crawled first, and some noises such as navigation bar, advertisement and logo usually exist in Web pages. They create difficulties to compute the relevance of Web pages.

4.2 Proposed Approach

A highly relevant region in a web page may be obscured because of low overall relevance of that page. Page segmentation transforms multi-topic web page into many single topic context blocks and hence improves its performance. These multiple-topic content blocks such as navigation panels, copyright and privacy notices, unnecessary images, and advertisements distract a user from the actual content and the performance reduces.

In this thesis, we present a method to divide the web pages into content blocks. This method uses an algorithm to partition a web page into content blocks with a hierarchical structure and partition the pages based on their pre-defined structure, i.e. the HTML tags. We have extracted content from HTML web pages and make the HTML tag tree of a block.

4.3 Content Block Partitioning From Web Pages

In a web page, the size of the region is variable. A big region covers the whole web page, but the size of smaller ones may be as small as 1/8 or 1/16 of the web page's total space. A content block is assumed to have a rectangle shape. A web page will be partitioned into blocks on the basis of *headings*. First we make the HTML tag tree of a block. Each HTML page corresponds to a tree where tags are internal nodes and the detailed texts, images or hyperlinks are the leaf nodes. One complete block comprises of a heading and its details. This block also includes images, links, text, tables related to that particular block only. When the pages are segmented into the content blocks, the relevant blocks are crawled to extract the relevant links from them. Then the relevancy value of each block is calculated separately and summed up to find the overall relevancy of the page. The relevancy of web page may be inappropriately calculated if the web page contains multiple unrelated topics, which, can be a negative factor. Instead of treating a whole web page as a unit of relevance calculating, we evaluate each content block separately.

4.4 Algorithm of content Block Partitioning:

A structure of node which is required to make a tree is:

```
struct node
{
    string nodename; // It contains name of node like of html tag node_name will
    be html.
    int nodeno;      // It contains node number which will be given according to
    BFS.
    string children;  // It contains node_no of child nodes.
    string content;   // It contains content like content of title or h2. Content of tags
    like html, head   are empty
}
```

1. Extract all tags like html, head, title body etc.

2. Fill the nodes of tree with node_no, children and content.
3. Now traverse the tree according to Breadth First Search.
4. If any heading tag like h1, h2, h3 etc occurs put that in a block.
5. Repeat until next heading tag is not arrived.
6. Put all content tags and their children like p, table, tr, td, th in same block.
7. If end of tree is reached.
8. End loop

```

Input (t: HTML parse tree according to BFS)
Procedure:
String[] heading={"h1","h2","h3","h4","h5","h6"};
Tt=t.
Block=0;// Refers to null block
Queue = root Tt
while (Queue is not empty)
{
if (heading.contains(Tt.nodename))
{
    Block = Block + 1.
    if (Tt has children)
        putTt and all children in Block.
    else
        putTt in Block.
}
else
{
    if(Tt has children)
        putTt and all children in Block.
    else
        putTt in Block.
}
}

```

```

<html>
<head>
<title>web Crawling</title>
</head>
<body>
<h1>Introduction</h1>
<p>A <a href="http://en.wikipedia.org/wiki/web_crawler">web crawler</a> is a program .....
<table>
<tr>
<th>Types of crawler</th>
<td>Focused Crawler, distributed crawler</td>
</tr>

<tr>
<th>approaches of crawler</th>
<td>Fish,<a href="http://www7.scu.edu.au/1849/com1849.htm">Shark</a><td>

</tr>
</table>

...</p>
<h3>what is<a href="http://en.wikipedia.org/wiki/web_search_engine">Search Engine</a>?</h3>
<p>By Search Engine in relation.....</p>
<h3>Types of Search Engines:</h3>
<p>The search engine belongs to 3..... <br/>
1. Those that are powered by robots (called crawlers, ants or<a href="http://en.wikipedia.org/wiki/spider"> spiders</a>).<br/>
2. Those that are powered by human submissions.<br/>
3. Hybrid search engines.<br/>
</p>
<h2>Focused Crawling</h2>
<p>One of the first focused web.....</p>
</html>

```

Figure 8. Illustration of a content block structure, a snippet of HTML pages

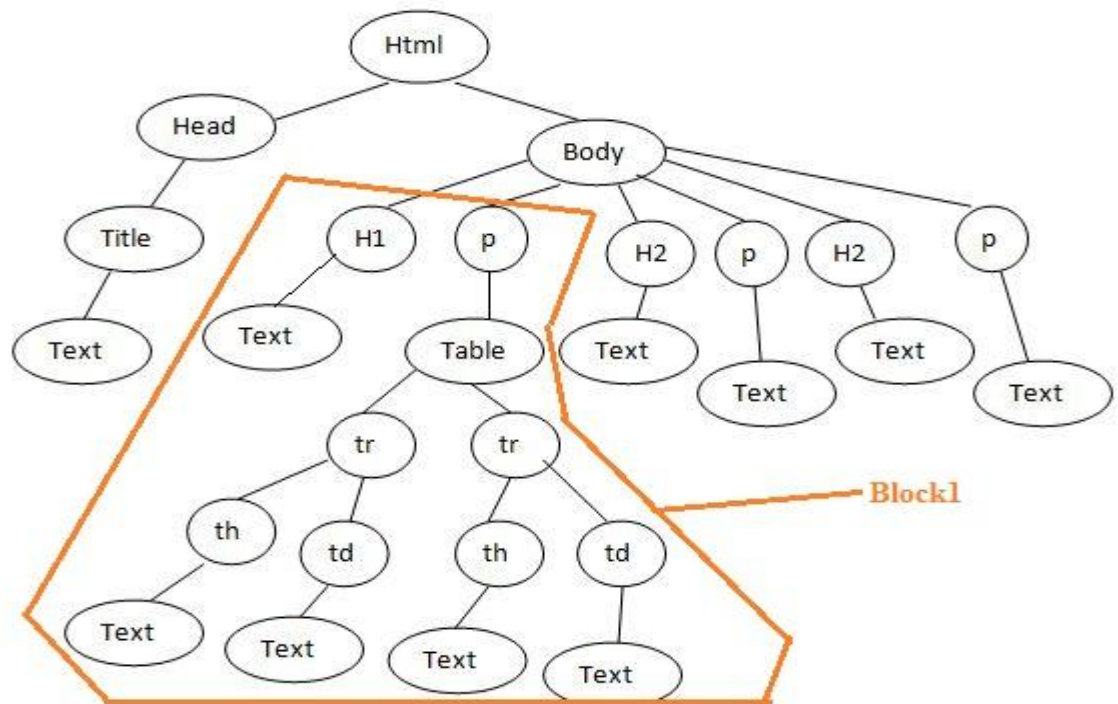


Figure 9. The tag tree of a block corresponds to an HTML source

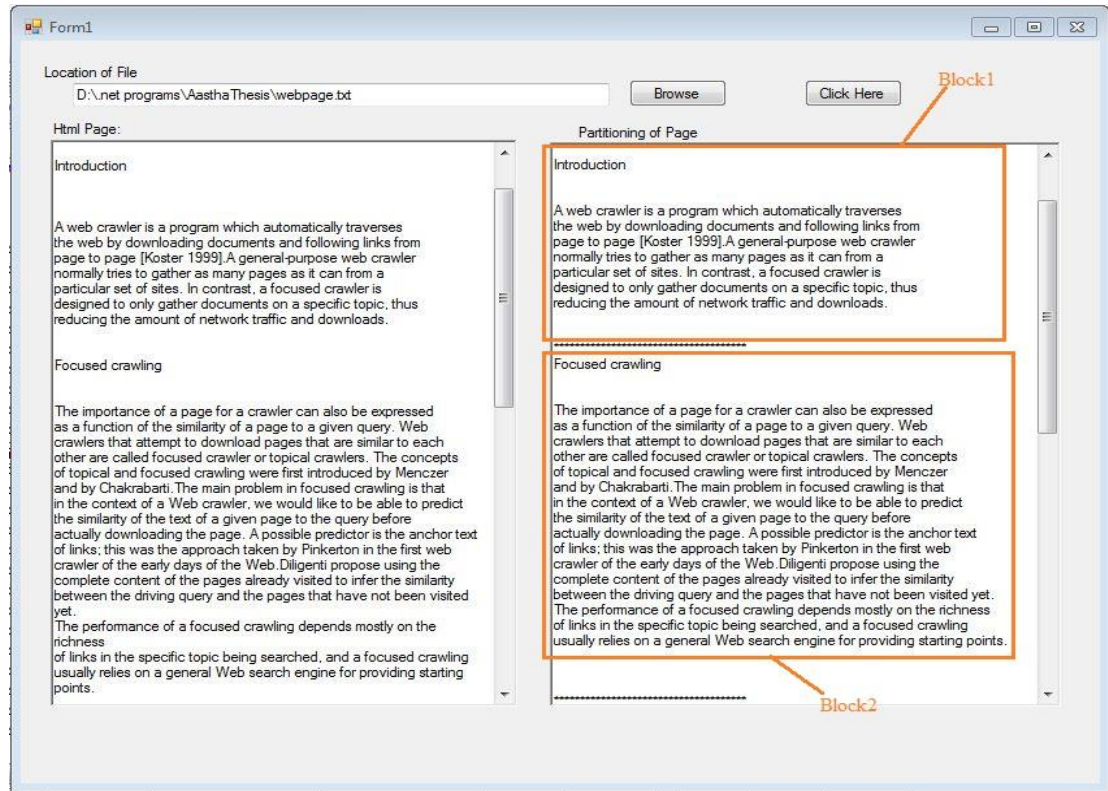


Figure 10. Partitioning of a web page into content blocks

After the content block partitioning of web pages, we have given architecture of a focussed crawler and explained all the terms and also the steps that are to be followed in focussed crawling after partitioning. But the method of crawling remains the same. But the difference is just that we have applied focussed crawling on large number of content blocks rather than the whole page. After this, we will calculate the relevancy value of each block and sum these values to find the relevancy of complete web page.

4.5 Focussed crawling procedure guided by content block partitioning

4.5.1 Topic Specific Weight Table Construction [4]

After the block partitioning, we decide whether a content block is relevant to the topic or not. First the retrieved block is parsed. Then, stop words such as "the" and "is" are eliminated. After that, words are stemmed and the term weight of each term which is in

topic table is calculated in this block. The term weight is computed using the following formula:

The term weights = {t1 t2 ti... t10} are computed as:

$$t_i = n_i / n_{\max}$$

Where n_i is the term occurrences in the web page and $n_{\max}$ is the frequency of the term with most occurrences.

4.5.2 Block Analysis

After calculating the weight of terms in block, we find out the relevancy score of block with respect to topic table. Relevancy score is calculated as: [4]

$$R(t, b) = \frac{\sum w_{kt} * w_{kb}}{\sqrt{\sum (w_{kt}) * (\sum (w_{kb}) * (w_{kb}))}}.$$

Here, t is the topic specific weight table, b is the block web page, and w_{kt} and w_{kb} are the weights of keyword k in the weight table and in the block of web page respectively. The range of Relevance (t , b) lies between 0 and 1. Based on relevancy score, we identify the block is relevant or irrelevant. If the relevance score of a block of web page is greater than relevancy limit specified by the user, then the URLs which are in that block are extracted for predicting the next crawling based on URL score.

4.5.3 URL Score Calculation

A hyperlink is a reference of a child web page that is contained in a parent web page. When the hyperlink is clicked on in a parent web page, then the browser displays child web page

4.5.4 Algorithm for URL Score Calculation [33]

Step 1: Extract LINKs from relevant block by "Link Extractor Tool."

Step 2: Find out all parent pages of each LINK by "Back Link Analyzer tool."

Step 3: Content block partition of each parent page.

Step 4: Identify blocks in each parent page in which specific LINK exists.

Step 5: Calculate the relevancy score of parent page block with respect to topic table terms.

/* We calculate the relevancy score of each block of each parent page in which this particular link exists. */

/*Here we are writing a statement for one block. */

Step 6: Calculate the weight of each topic table terms in block.

Step 7: Extract the weight value of each topic table's terms in particular block.

Step 8: Calculate the relevancy score of parent page block with respect to topic.

Step 9: for i = 1 to 10

$$R(t, b) = \frac{\sum wkt \cdot wkb}{\sqrt{\sum (wkt) \cdot (wkt) \cdot \left(\sum (wkb) \cdot (wkb) \right)}}$$

Step 10.

/*Repeat step 3 to 10 until we find out all parent pages block relevancy score. */

Step 11: Calculate average parent page block relevancy score.

/* we extract all parent page blocks' relevancy score and then find out average of this relevancy score.*/

Step 12: $R(t, 1)$ = average parent page block relevancy score.

/* $R(t, 1)$ is the relevance score of link with respect to topic. */

Step 13: $\text{Score}(u) = \mu R(t, b) + (1 - \mu) R(t, 1)$.

/* $R(t, b)$ is the relevance of block with respect to topic. */

/*Score (u) is score of unvisited URL and μ is a parameter which can be adjusted in experiments. The initialization of μ is set to 0.5. */

4.6 Dealing with Irrelevant Pages

Sometimes it can happen that irrelevant pages can be linked to relevant ones. We skip all the irrelevant pages and do not parse them assuming that they will lead to a dangling node (having nothing relevant). But actually it is not so. The irrelevant pages can also lead to the relevant ones. A technique called tunneling is described for traversing the irrelevant pages to reach relevant ones. Let $n_1, n_2, \dots, n_k$ be irrelevant web pages with links n_i pointing to n_{i+1} ($1 \leq i \leq k-1$), and p be a relevant page pointed by n_k , then $n_1, n_2, \dots, n_k \rightarrow p$ is defined as a tunnel. The process of traversal $n_1, n_2, \dots, n_k$ to reach p is called **tunneling**.

For the solution of this problem, we have given an algorithm that is applied on irrelevant blocks that will lead to some relevant links. The principle of this algorithm is to go on crawling upto a given maxLevel from the irrelevant page.

4.6.1 Pseudo code of algorithm [4]

```
1. If (page is Irrelevant)
2. {
3. Initialize level = 0;
4. url_list = extract_urls (page);
   // extract all the urls from the page
5. for each u in url_list {
6. compute Link Score(u) using equation (4);
7. IrrelevantTable.insert(u, LinkScore (u), level);
   // insert u into irrelevant table with linkscore and level value
   }
8. reorder IrrelevantTable accord to LinkScore(u);
9. while ( IrrelevantTable.size > 0)
10. {
11. get the url with highest score and call it Umax ;
12. if ( Umax.level <= maxLevel)
13. {
14. page = downloadPage(Umax);
   // download the URL Umax
15. calculate relevance of the page using equation (3);
16. if ( page is relevant) {
17. RelevantPageDB = page;
   // put page into relevant page database
18. if ( Umax.level < maxLevel) {
19. level ++ ;
```



```

20. url_list = extract_urls(page);
21. for each u in url_list {
22. compute LinkScore(u) using equation (4);
23. IrrelevantTable.insert(u, LinkScore(u), level); }
24 reorder IrrelevantTable accord to LinkScore(u); }
25.} else {
26. for each u in IrrelevantTable {
27 if (LinkScore(u) <= LinkScore(Umax)
    && u.level == Umax.level)
28. u.level ++; }
29.}
30.}
    }
}

```

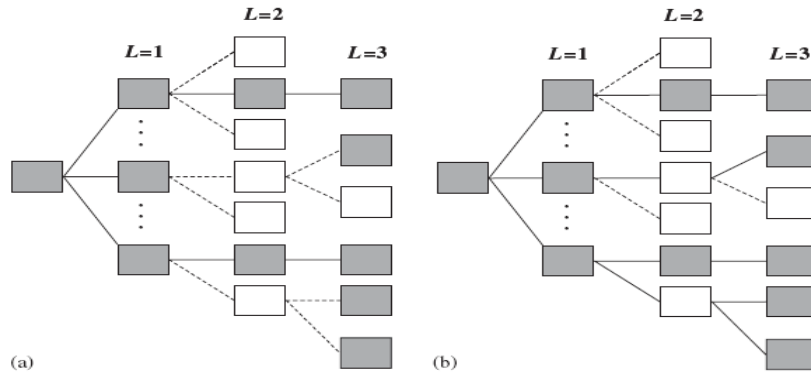


Figure 11. Illustration of the path focused crawling when encountering irrelevant page, in which some content blocks may be relevant: (a) focused crawling process without tunneling and (b) focused crawling process with tunneling.

4.7 Results and Discussions

A highly relevant content in a web page can be obscured in the whole page because there exist some irrelevant topics also which reduces the overall relevancy of the web page.. Accordingly, partitioning the web pages into smaller units will significantly improve the focused crawling performance. Sometimes, also it can happen that we

ignore the irrelevant pages but if we traverse them it may lead to some relevant content. So in order to maximize efficiency and to improve the performance we have retrieved the useful web pages and divided the page into content blocks. In our proposed method of partitioning the web pages into blocks on the basis of headings gives an advantage over conventional block partitioning is that we divide the blocks which include a complete topic. The heading, content, images, links, tables, sub tables of a particular topic is included in one complete block. As compared to previous methods of partitioning, this method is more appropriate because in other methods, sub tables of a table are considered to be the other block. But it is not so. On the basis of headings, there is an appropriate division of pages into blocks. We used conventional classifier to decide whether the page is relevant. *Harvest rate*, *Target recall* and *Target length* are used to evaluate the results

4.7.1 Performance metrics

The output of a crawler is a sequence of pages crawled. Any evaluation of crawler performance is based on this output. We can estimate the *precision* and *recall* of a crawler after crawling n pages. The *precision* would be the fraction of pages crawled that are relevant to the topic and *recall* would be the fraction of relevant pages crawled. However, the relevant set for any given topic is unknown in the web, so the true recall is hardly to measure. The most crucial evaluation of focused crawling is to measure the rate at which relevant pages are acquired, and how effectively irrelevant pages are filtered out from the crawl. Similarly, defined as the rate at which crawled pages satisfy a given predicate, ‘harvest rate’ which is a running average, over different time slices of the crawl, of page relevance assessed using classifiers. *Harvest rate* will be used as an estimate of precision and *target recall* will be used as an estimate of recall. For testing the performance of tunneling, we also present a *target length* metrics.

Harvest rate: estimates the fraction of crawled pages that are relevant to a given topic. We make this decision by using a set of SVM evaluation classifiers instead of manual relevance judgment, which is costly. We further modified training set as input. The SVM evaluation classifiers are trained on a larger set from ODP, which are more ‘informed’ about the topic. To explain, for each topic, we take one classifier at a time and train it using the pages corresponding to the entire ODP relevant set (instead of just

the seeds) as the positive examples. The negative examples are obtained from the ODP relevant sets of the other topics. The negative examples are again twice as many as the positive examples.

$$\text{harvest rate} = \text{relevant pages} / \text{pages downloaded}$$

Target recall: is an estimate of the fraction of relevant pages that are fetched by a crawler. As described earlier, true recall is hardly to measure since we cannot identify the true relevant set for any topic over the Web. So, we treat the recall of the target set, i.e. target recall, as an estimate of true recall for different techniques comparing. If targets are a random sample of the relevant pages on the Web, then we can expect target recall to give us a good estimate of the actual recall. We assume that the target set T is a random sample of the relevant set R . Therefore, for recall, $|C(t) \cap R|/|R| \approx |C(t) \cap T|/|T|$. The target recall, after first crawling t pages for a given topic is computed a

$$R(t) = |C(t) \cap T| / |T|$$

where $C(t)$ is the set of first t pages crawled, T is the set of targets, and $|T|$ is the number of targets.

Target length: The metrics are presented for tunneling performance. Target length L is the distance from seed URL to target relevant web page.

5.1 Conclusion

Web crawlers are the program that uses the graphical structure of the Web to move from page to page. A focused crawler is a crawler that targets a desired topic and gathers only a relevant Web page which is based upon predefined set of topics and do not waste resources on irrelevant web pages. *Best-first search* is the most popular search algorithm used in focused crawlers. In best-first search, URLs are not just visited in the order they are present in the queue; instead, some rules are applied to rank these URLs. But we see there are multiple URLs and topics on a single web page. So the complexity of web page increases and it negatively affects the performance of focussed crawling and the overall relevancy of web page decreases. A highly relevant region in a web page may be obscured because of low overall relevance of that page. The overall performance can be improved by segmenting the web pages. Page segmentation transforms the multi-topic web page into several single topic context blocks. This method is known as content block partitioning. We will present an algorithm how to efficiently and accurately divide the web page into content blocks and then we will apply focussed crawling on the content blocks.

In this thesis, we have briefly discussed about Internet, Search Engines, Web Crawlers, Focussed Crawlers and Block Partitioning of Web pages.

In this thesis an approach is proposed to partition the web pages into content blocks. Using this approach we can partition the pages on the basis of headings and preserve the relevant content blocks. Therefore a highly relevant region in a low overall relevance web page will not be obscured. In our proposed method of partitioning the web pages into blocks on the basis of headings gives an advantage over conventional block partitioning is that we divide the blocks which include a complete topic. The heading, content, images, links, tables, sub tables of a particular topic is included in one complete block. As compared to previous methods of partitioning, this method is more appropriate because in other methods, sub tables of a table are considered to be the

other block. But it is not so. On the basis of headings, there is an appropriate division of pages into blocks. After calculating the relevancy of different regions it calculates the relevancy score of web page based on its block relevancy score with respect to topics and calculates the URL score based on its parent pages blocks in which this link does exist.

5.2 Future Scope

Future scope can include:

- Code optimization
- Blocks can be divided on some other way in which the overall complexity of web page decreases and also it will be easier to apply focussed crawling on these pages.
- Also the time complexity must be reduced because crawler efficiency not only depends on to retrieve maximum number of relevant pages but also to finish the operation as soon as possible.

References

- [1] Arvin Arasu, Junghoo Cho, Andreas Paepcke: “Searching the Web”, Computer Science Department, Stanford University.
- [2] Mr.Ravinder Kumar, Sandeep Sharma: “Web Crawling Approaches in |Search Engines” CSE Department, Thapar University ,2008.
- [3] Brian Pinkerton,"Web Crawler: Finding what people want ", Doctor of Philosophy University of Washington, 2000.
- [4] Mr.Ravinder Kumar, Ravikiran Routhu: “Enrichment in Performance of Focussed Crawlers” CSE Department, Thapar University ,2010.
- [5] Fabrizio Silvestri, "High performance issues in web search engines: Algorithms and Techniques", PhD: Thesis: TD 5/04 ,available at :http://hpc.isti.cnr.it/_silvestr May 2004.
- [6] Paul De Vrieze, "Improving search engine technology, "Master thesis, 7-March-2002
- [7] Gautam Pant, Padmini Srinivasan, and Filippo Menczer: “Crawling the Web” available at- << <http://dollar.biz.uiowa.edu/~pant/Papers/crawling.pdf> >>
- [8] Lovekesh Kumar Desai,"A distributed approach to crawl domain specific hidden web", georgia state university, 2007.
- [9] S. Chakrabarti, M. van den Berg, and B. Dom. Focused crawling: “A new approach to topic specific Web resource discovery”. Computer Networks.

- [10] Mark najork and Allan Heydon, "High Performance web crawling", SRC Research Report 173, published by COMPAQ systems research centre on Sep 26, 2001
- [11] Sergey Brin and Lawrence Page, "The anatomy of a large-scale hyper textual Web search engine", In Proceedings of the Seventh International World Wide Web Conference, pages 107–117, April 1998.
- [12] S. Chakrabarti. "Mining the Web". Morgan Kaufmann, 2003.
- [13] Chang C, KayedM, Girgis MR and Shaalan KF,"A survey of Web Information Extraction systems", IEEE Transactions on knowledge and engineering.2006.
- [14] Kraft R and Stata R,"Finding buying guides with a web carnivore", First latin American Web Congress ,pages 84-92.,2003
- [15] Pant G, Johnson J and Giles C.L," Panorama: Extending digital libraries with topical crawlers", Proceedings of the 4th ACM/IEEE-CS Joint Conference on Digital Libraries.2004,pages 142-150 [14] Kraft R and Stata R,"Finding buying guides with a web carnivore", First latin American Web Congress ,pages 84-92.,2003
- [16] S. Chakrabarti, B. Dom, P. Raghavan, S. Rajagopalan, D. Gibson, and J. Kleinberg, "Automatic resource compilation by analyzing hyperlink structure and associated text," in Proc. 7th World Wide Web Conference, Brisbane, Australia, 1998.
- [17] K. Bharat and M. Henzinger, "Improved algorithms for topic distillation in hyperlinked environments," in Proceedings 21st Int'l ACM SIGIR Conference., 1998.
- [18] J. Kleinberg, "Authoritative sources in a hyperlinked environment." Report RJ 10076, IBM, May 1997, 1997.

- [19] B. Pinkerton, "Finding what people want: Experiences with the web crawler," in Proceedings of the First International World-Wide Web Conference, Geneva, Switzerland, May 1994.
- [20] F. Menczer, G. Pant, and P. Srinivasan, "Topical web crawlers: Evaluating adaptive algorithms," ACM Transactions on Internet Technology (TOIT), Vol. 4, no. 4, pp. 378–419, 2004.
- [21] A. Grigoriadis, "Focused crawling using reinforcement learning," Master's thesis, The University of Edinburgh, 2003.
- [22] M. Diligenti, F. Coetzee, S. Lawrence, C. L. Giles, and M. Gori, "Focused crawling using context graphs," in 26th International Conference on Very Large Databases, VLDB 2000, Cairo, Egypt, 2000, pp. 10–14.
- [23] Z. Guan, C. Wang, C. Chen, J. Bu, and J. Wang, "Guide focused crawler efficiently and effectively using on-line topical importance estimation," in Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2008, Singapore, July 20-24, 2008, 2008, pp. 757–758.
- [24] Menczer F, Pant G and Srinivasan P," Topical Web Crawlers: Evaluating Algorithms "ACM Transactions on Internet Technology (TOIT), Nov, 2004.
- [25] F. Menczer and R. K. Below. Adaptive retrieval agents: "Internalizing local context and scaling up to the Web". Machine Learning, 39(2{3):203{242, 2000.
- [26] F. Menczer, G. Pant, and P. Srinivasan. Topical web crawlers: Evaluating adaptive algorithms. To appear in ACM Transactions on Internet Technologies, 2003. <http://dollar.biz.uiowa.edu/Papers/TOIT.pdf>.

- [27] P. DeBra and R. Post, Information retrieval in the worldwide web: making client-based searching feasible, in: Proc. of the 1st International World Wide Web Conference, Geneva, Switzerland, 1994.
- [28] M. Hersovici, M. Jacovi, Y.S. Maarek, D. Pelleg, M. Shtalheim and S. Ur, "The Shark-Search algorithm — an application: tailored Web site mapping," in: 7th World-Wide Web Conference, April, 1998, Brisbane, Australia, online at <http://www7.scu.edu.au/programme/fullpapers/1849/com 1849.htm>
- [29] Bing liu, "web data mining", Springer-Verlag Berlin Heidelberg, 2007
- [30] Junghoo Cho, Hector Garcia-Molina: "Parallel Crawlers", 7–11 May 2002, Honolulu, Hawaii, USA.
- [31] Yang Yongsheng, Wang Hui, " Implementation of Focussed Crawler" COMP 630D course Project Report, 2000
- [32] Li Wei-jiang, Ru Hua-suo and Zhao Tie-jun, Zang Wen-mao, "A New Algorithm of Topical Crawler", Second International Workshop on Computer Science and Engineering, 2009.
- [33] Debashis Hati, Amritesh Kumar "Improved Focused Crawling Approach for Retrieving Relevant Pages Based on Block Partitioning". 2nd International Conference on Education Technology and Computer (ICETC) 2010.

