1.1 Web Search Process

The World Wide Web is an ever expanding source of information that reflect the interests and views of diverse global society. Many individuals, organizations, and institutions describe their interests, mission, activities, future plans, and research on their Web sites. Online digital libraries, tutorials, and white papers provide access to professional and technical information. These are just a few examples of the diverse information available on the Web. The important characteristic of the Web is that it is not a simple collection of documents and pages but an interconnected collection. The connections between pages that are made through hyperlinks facilitate manual navigation or surfing over the Web. Hence, the Web can be seen as a graph with pages forming its nodes and hyperlinks forming its edges. Since hyperlinks have a source page and a destination page, the edges of the Web graph have dimensions.

The Web is a constantly changing entity. The contents of Web pages are being changed; new pages are being added to the Web while many others are being disappeared from it over the time. The hyperlinks between pages may also vary with time. This dynamism of the Web is largely uncontrolled. The lack of control over the Web also brings up the issue of information quality and reliability. Sometimes the pages with little or no information are retrieved in response to users’ queries. At other times, they contain unreliable or incorrect information. On other occasions we find very valuable and accurate information. Hence, the large size dynamism,
and uncontrolled nature of the Web offer new challenges for information handling, retrieval and usage.

General purpose search engines like Google and Yahoo! are popular tools for information search over the Web. A search engine captures and updates a snapshot of portion of the Web by automatically downloading pages at regular intervals. At any given time a search engine has a set of previously downloaded pages that it can use to approximate the current state of the Web. Based on these approximations search engine attempts to provide relevant result to diverse user queries. The retrieval solution adopted by first generation search engines depend only on the information provided at the nodes of the Web graph. They store the contents of the collected pages as an inverted index for fast lookup at query time. These search engines made no use of the information captured through the edges of the Web graph for page representation and retrieval. As mentioned earlier, the Web has pages with varying quality and reliability of information. This creates the problem of irrelevant results from search engines. As the size of the Web grew from hundreds of thousands to hundreds of million pages, the problem of irrelevant results left search engine’s users with significant overhead of filtering out irrelevant results.

Google pioneered an innovative technique of ranking search engine results based on graph structure of the Web that not only considered the content in pages but also their hyperlinks. Google displays the relevant result in decreasing order of their estimated popularity. One way to estimate popularity of a given page is to count the number of pages that
have hyperlinks to given page. Google's popularity measures extend this idea through a more sophisticated algorithm based on the graph built from the downloaded pages. This approach tackles the information overload by directing the user's attention to the most popular sources of information.

It is important to note that the general purpose search engines strive for as broad coverage of the Web as possible. These generic search engines attempt to download an ever increasing fraction of Web pages so that they can cater to a diverse user base. One may question the effectiveness of this one-size-fit-all technique in catering to specific information needs of a user, a community, or an organization. When information needs are specific, possibly requiring in-depth research, there is a need to build niche search engines. There is also growing interest in knowledge discovery systems that typically rely on topical collections constructed by focused crawlers.

The general Web search process has two main parts: off-line and on-line as shown in Fig. 1.1. The offline part is responsible for consistently downloading the sub part of the Web to build and update the collection of pages. This part is called by the search engine periodically. The collection of the pages is then used for constructing searchable index. Whenever the search engine receives a query the online part makes use of the searchable index to retrieve some of the pages matching the query. These documents are further sorted as according to their relevancy to the query terms. Web pages may appear in any format like PDF documents, plain text, HTML tags, etc. The first step towards indexing the Web pages is to extract a logical view from the documents. Unordered set of words from the pages is
used most popularly for extracting this logical view from Web pages. Many text normalization techniques like tokenization, stop words removal, stemming etc. are applied on the text extracted from the Web pages. Tokenization is the process of making individual tokens (words) from the text. Stop words like is, am, are, where, what etc possesses only a little semantic meaning and hence can be removed from the text.

Fig. 1.1: Web search process
Stop words removal improves space and time requirements of the overall crawling approach because stop words appear with a high frequency and hence consume large space for storing and time for processing. For retrieving morphological root of a word stemmer is used. Stemming extracts the morphological root of every word. In global search engines, the main problem with stemming is that it is language dependent, and while rule-based stemming is applied for English language, a dictionary-based approach is used for Spanish language.

1.2 Focused Collections

Focused collections are useful in the process of knowledge discovery. The huge collection of hundreds of billions of pages is more prone to the problems of polysemy and synonymy than the focused collection. The word “power” may appear in a large number of pages in a general collection and is having multiple meanings, but the number of possible meanings can be reduced by considering the “automobile” domain.

Focused collections are more likely to provide homogeneous contexts which can help reduce the space and complexity of information search strategies. The words “robots”, “crawlers”, and “spiders” may not appear to be synonyms in a generic collection but in a focused collection that has pages about Web crawler technology, these words would be considered similar.
A focused collection contains information associated with a particular field. This collection can be used to find connections and relationships among various unknown concepts. Search engines use the focused collections to adhere to individual users or communities that have some common focused interests. These search engines are more scalable than the others by citing the case that they need to cover only a small portion of the Web. The smaller portion is relatively easier to maintain and update than the larger one. These focused collections are smaller in size and rich in information about the topic of interest, and hence can serve the user in a better way than the generic collections. After understanding the need for focused collections, the question that arises is: How to generate these kinds of focused collections? A focused crawler can be seen as a solution for this question.

1.3 Crawling Policies

For maintaining its index a search engine makes use of a crawler. The cost of crawling and indexing is overcome by serving many number of users' queries with the help of same collection and same index. Web crawlers are programs that traverse the Web, page by page and link by link to download them. Web crawlers are also called robots, fish, wanderers, spiders, and worms. The general crawler starts from a set of seed pages and traverses the hyperlinks present within the pages to process the other pages. The process reiterates with the new pages until the crawler frontier is not empty or the requested number of links are not crawled.
Fig. 1.2: General crawler

Flow of a basic sequential crawler is given in Fig. 1.2. The crawler is started with a number of seed URLs which are added to the crawl frontier. The crawler picks a URL from the frontier, fetches the URL through HTTP from the Web, parses the fetched page and then inserts the links found on the page to the frontier. The crawler can stop when fixed number of Web pages is crawled or when a fixed crawling depth is achieved. These conditions depend upon the type of crawler.

The crawl graph for a general crawler is shown in Fig. 1.3. The seed URLs can be seen as the start nodes of the crawl-graph. From these start nodes, the Web-graph is traversed by the crawling loop and
simultaneously, the crawl-graph is built. As the Web-graph is not an acyclic graph, some ways to detect these cycles must be provided to prevent the crawling loop to continue endlessly. The frontier can be seen as a queue that contains the unseen URLs for future crawling. The crawler queue can also be seen as open list of unexpended nodes of the crawl graph.

A FIFO queue data-structure can be used to store the URLs. With a FIFO queue, the crawler will blindly visit all URLs in a breath-first fashion. Another possibility is to use a priority-queue to store the URLs. In that case, the crawler uses a heuristic to compute a score to every page. The page with the highest score will be visited next. In that way, the crawler traverses the graph in a best-first manner.

![Crawl graph for a general crawler](image)
A Web crawler can be seen as a combination of selection policy, re-visit policy, politeness policy, and parallelization policy.

**Selection Policy:** Selection policy means the criteria that decides whether a page is to be downloaded or not by the crawler. Given the exponential growth of the Web, no search engine is able to index the complete Web. A Web crawler downloads the Web pages throughout all the times and still it is able to reach to only a portion of the whole Web and each crawler try to download as many pages as possible. The downloaded pages are expected to contain only highly relevant pages and not any random set of pages.

The importance of a page is its inherent quality, and the links contained within it and its visits determine this importance. The selection policy is the real challenge for any focused crawler as it works upon some heuristics for deciding which URL to follow and which URL to discard without actually downloading the URL.

**Re-visit Policy:** The ever-increasing size and dynamism are the main characteristics of the Web. Its state is changing every fraction of time. Crawling a portion of the Web takes a significant amount of time. While reaching to the end of crawling for a portion, the state of that portion may have changed i.e. many of the pages from the crawled portion may be deleted, modified or new pages may have been uploaded. The search engine must be able to respond to the user with latest and consistent contents all the times. Thus it must have to consider the dynamism of the Web while serving the user. This dynamism is taken care by the re-visit
policy. Re-visit policy makes use of freshness and age of the Web page to deal with the dynamism.

**Freshness:** It measures the accurateness of the contents of the actual Web page lying on the Web with the copy of the page lying with the crawler. It is a binary value and is given by

$$ F_p(t) = \begin{cases} 1 & \text{if } p \text{ is equal to the local copy at time } t \\ 0 & \text{otherwise} \end{cases} $$  

(1.1)

**Age:** It is a measure of the actual age of the page i.e. the current time minus the time at which the page in its current form came to the repository or the time at which the page was modified last. It is given by

$$ A_p(t) = \begin{cases} 0 & \text{if } p \text{ is not modified at } t \\ t - \text{modification time of } p & \text{otherwise} \end{cases} $$  

(1.2)

The purpose of the crawler is to keep average freshness of the Web page as high as possible and average age as low as possible. The crawler has to look for pages which are fresh enough and also to look as many pages as possible that are not outdated.

**Politeness policy:** This policy is concerned about the behavior of the Web crawler towards the Web server. If the crawler is sending multiple requests to a server for downloading large files at the same time then the server will not be able to serve its best to other users as well as to the crawler. Also the crawling costs that include the cost of network resources
will increase as the multiple crawlers are going to consume a considerable amount of network bandwidth. A badly written crawler can results in crashing down a server by putting multiple number of requests to the server at the same time. The costs of using Web crawlers include:

a. Network resources, as crawlers require considerable bandwidth and operate with a high degree of parallelism during a long period of time.

b. Server overload, especially if the frequency of accesses to a given server is too high.

c. Poorly written crawlers, which can crash servers or routers, or which download pages they cannot handle.

d. Personal crawlers that, if deployed by too many users, can disrupt networks and Web servers.

One of the solution opted by the Web server administrators is the use of robot exclusion protocol. With the help of robot exclusion protocol portions of the Website which are not accessible to a general crawler can be marked and hence the general crawlers are denied access to these portions.

**Parallelization Policy:** The parallelization policy allows the crawler to run multiple processes in parallel. It results in maximizing the download frequency. Avoiding downloads of the same page is one of the main challenges for the crawlers. Different crawler architectures opts for different mechanism for avoiding the repeated downloads.
1.4 Focused Web crawler

A focused crawler downloads Web pages that are relevant to a particular topic and avoid downloading all others. It predicts the probability that a link to a particular page is relevant before actually downloading the page. Relevant pages are sent to content indexing and their contained URLs are added to the crawl frontier, pages that fall below a relevance threshold are discarded by considering them as irrelevant to the domain. The flow of a general focused crawler is depicted in Fig. 1.4.

A general focused crawler starts by initializing the crawl frontier with seed URLs. Seed URLs are the links which are considered to be relevant to the topic for the focused crawling. The seed URLs are fetched from WWW and preprocessed for tokenization, stop words removal etc. After preprocessing, the similarity of the page to the domain, or the class of the page (relevant or irrelevant) is determined by using some similarity measures like $tf – idf$, cosine similarity or some classifiers like Naïve Bayes, Support Vector Machines (SVM), Neural Networks etc.

Once a page that belongs to the relevant class or having the similarity score greater than some threshold value is found, all the URLs contained within the page act as the future candidates for the results containing the topic relevant pages, and hence are inserted into the crawl frontier. The Web graph traversal behavior of a focused crawler is given in Fig. 1.5 which shows that only the relevant pages (white nodes) are expanded further for future crawl.
The non relevant pages (grey nodes) are discarded for future crawl, the black node are the ones lying in the crawl frontier with their respective position as according to the similarity of the link to the domain. The crawl graph clearly indicates the saving in resources by avoiding some irrelevant portions of the Web.
Focused Web crawler can be used for specialized search engines. Search engines make use of the crawler to collect the Web pages accessible on the Web for indexing. The search index is further used to serve the end user. In case of a general crawler the deep association between the search engine and the crawler is not needed because they may be supposed to work individually i.e. the crawler work for collecting as many page as possible without consultation to the search engine for any crawl path related queries, on the other hand the function of the search engine is to index the pages coming out of the crawler. But when it comes to the focused crawler the crawler and the search engine are associated with a large degree to serve the user with the pages which are related to the topic domain.

![Crawl graph of a focused best-first crawler](image-url)

**Fig. 1.5: Crawl graph of a focused best-first crawler**
The pages produced by the crawler are further to be graded good or bad by the users through the search engine, and in return the focused crawler tries to improve the quality of the pages being downloaded for indexing. This symbiosis helps to generate a quality collection of pages.

Digital libraries may also use the focused crawler. Many technical, professional, or research documents have quality information that can be used to find associated resources on the Web. Due to its largely uncontrolled nature, the Web in general can be expected to be of lower quality when compared with the more controlled material available as digital libraries, research proposals, white papers, and other documents that go through peer review or editing. A tool that effectively connects the “quality” information in digital documents, with appropriate resources within the large “quantity” of information available on the Web will likely be of value to research and decision making.

Contents in a well formatted document can be used to automatically discover Web resources associated with it. For example, the title, author names, keywords, noun phrases, and references within a document can be used to query a search engine. Using the “best” results from the search engine; a focused crawler can be trained to build a focused collection that could give wide ranging information related with the original document. The crawled pages can also be clustered to discover key Web communities and how they relate to each other.

Focused crawling can also be used in knowledge mining The objectives, current functioning state, assets, revenue generation trends,
product details etc. of a lot of organizations is available on the Web. The information available may change very frequently by the organizations. All the peer organizations are always interested in the information available regarding their competitors to grow themselves by putting their best effort. All that information related to a certain category can be generated by using a focused crawler. There are many ways in which focused crawlers can leverage context and domain specific knowledge to build and maintain an up-to-date topical collection about such organizations. Such collections can then be mined using text mining tools. For example, based on the collection we may test a hypothesis such as “The organization x in Africa have a relationship with organization y in Europe”.

Topic specific portals such as Websites that index the research publications, Websites that contains the information related to the share market worldwide etc. may make use of focused crawlers. One of such kind of application is developed by McCallum et al. [1] to collect and maintain the research papers. Their approach finds the crawling policies that generate short term and long term benefits. The weight of the benefit depends upon the distance of the page from the current page. A page which can be reached with a fewer number of links is preferred over another page that is reachable with a greater number of links. It also follows the fact that the lexical relatedness between pages decrease with the increasing distances. The reinforcement learning crawler is trained with a number of paths known to reach to the related pages. The trained crawler
Focused Crawlers: State of the Art

is used to predict the benefit of crawling a link for reaching to the target page.

The site map can be seen as a pictorial or graph representation of the crawled Web. The breadth first crawling is considered to be a good technique for site map building. It traverses the Web until a specific depth is not reached or the required number of pages is not crawled. If the site map for the pages related to a particular topic is desired then this approach fails as it tries to cover the Web exhaustively independent of the type of pages being encountered. Mapuccino [2] corrects this by using Shark-Search. Shark-search is used to keep the crawler focused only to the relevant portion of the Web, which are shown as highlighted nodes of the site-map.

During browsing the user is helped by Letizia[3] to find the relevant pages. From the surfing behavior of the user Letizia learns the user’s interests. With the help of this it tries to reach out to more pages which can be of the interest of the user. It covers the Web in a breadth first manner. The user is not disturbed at all while learning the interests, but is suggested with a number of pages that may be relevant to him.

MySpiders[4] is an implementation of the InfoSpiders and Naïve best first algorithm. It is available online. Multiple crawlers are activated upon receiving a query. Once the crawler finds a related page it is displayed to the user. The result can be browsed while the crawling process ends. Each activated crawler is having an independent crawler queue. Naïve best first approach is applied in the same way. The choice of
the crawling algorithm is taken from the user with the help of a Java applet.

Focused crawling and general crawling algorithms are applied to various other applications. Business intelligence can also be seen as the beneficiary of the focused crawling. An organization may use a focused crawler to gather the information required to stand with the others after a thorough analysis. Crawling in general and focused crawling in particular is being applied for various other applications, many of which do not appear as technical papers. Focused crawler can also be used in biomedical science to collect the information related to a particular gene structure [5].

1.5 Surfing and Web Crawling

Web surfing [6] is possible due to the fact that most pages link to similar pages. Menczer [7] estimated the semantic similarity from a combination of link and content analysis. Menczer [8] provide interesting insights into the relationship between content similarities and relatedness among Web pages. He finds that both content and links provide a weak yet significant signal about the relatedness of Web pages. Another aspect that makes Web surfing possible is that the contextual information found around the hyperlink is suggestive of the destination page. For example, words within a hyperlink or around it tend to describe some of the aspects of the destination page. Not many would use the Web if these properties were not by and large true. Feasibility of a topical crawler relies on the same properties that make Web surfing possible. In fact, Web crawling can
be seen as automated surfing with the added availability of large memory and fast processing. However, unlike a computer, a human surfer is very perceptive and effective in identifying the context of a hyperlink even if it varies in size (number of words), positioning, or word usage. So the question arises: Can we train a topical crawler to exploit contextual cues in a manner similar to a human surfer? In other words, can a crawler select appropriate segments of text and estimate by their content the probability that a hyperlink leads to a relevant page? This remains the basic question while working on any focused crawling technique and is also treated as the important evaluation parameter for the different focused crawling strategies.

1.6 Crawler Evaluation Parameters

Precision, Recall and F-measures are used by most of the researchers to evaluate any information retrieval system. These parameters are also used for the focused crawler’s evaluation. All these are given below:

**Precision:** Precision is the probability that a (randomly selected) retrieved document is relevant or is the average probability of relevant retrieval and is given by

\[
Precision = \frac{\text{Number of related pages retrieved}}{\text{Total number of pages retrieved}}
\]  

(1.3)
Recall: Recall is the probability that a (randomly selected) relevant document is retrieved in a search or is the average probability of complete retrieval and is given by

\[
\text{Recall} = \frac{\text{Number of related pages retrieved}}{\text{Total number of related pages present}}
\]  

(1.4)

The random selection refers to a uniform distribution over the appropriate pool of documents; i.e. by randomly selected retrieved document, we mean selecting a document from the set of retrieved documents in a random fashion.

F-measure: A measure that combines precision and recall is the harmonic mean of precision and recall and is called F-measure or balanced F-score and is given by

\[
F = \frac{2 \times \text{Recall} \times \text{Precision}}{\text{Recall} + \text{Precision}}
\]  

(1.5)

This is also known as the $F_1$ measure, because recall and precision are evenly weighted.

Some of the benchmark studies in the field of crawlers and focused crawlers are given below:

Web crawlers are almost as old as the Web itself. In 1993, Matthew Gray implemented the World Wide Web Wanderer [9]. The Wanderer was written in Perl and ran on a single machine. It was used until 1996 to collect statistics about the evolution of the Web. Moreover, the pages crawled by the Wanderer were compiled into an index (the “Wandex”), thus giving rise to the first search engine on the Web. In December 1993, three more
crawler-based Internet Search engines became available: JumpStation (implemented by Jonathan Fletcher; the design has not been written up), the World Wide Web Worm [10], and the RBSE spider [11].

1.7 WebCrawler

WebCrawler [12] and MOMspider [13] appeared in April 1994. This first generation of crawlers identified some of the defining issues in Web crawler design. For example, MOM-180-spider considered politeness policies: It limited the rate of requests to each site, it allowed Web sites to exclude themselves from purview through the nascent robots exclusion protocol [14], and it provided a “black-list” mechanism that allowed the crawl operator to exclude Websites. WebCrawler supported parallel downloading of Web pages by structuring the system into a central crawl manager and 15 separate downloading processes. However, the design of these early crawlers did not focus on scalability, and several of them (RBSE spider and WebCrawler) used general-purpose database management systems to store the state of the crawl. Even the original Lycos crawler [15] ran on a single machine, was written in Perl, and used Perl’s associative arrays (spilt onto disk using the DBM database manager) to maintain the set of URLs to crawl. Few years saw the arrival of several commercial search engines (Lycos, Infoseek, Excite, AltaVista, and HotBot), all of which used crawlers to index tens of millions of pages; however, the design of these crawlers remains undocumented.
1.8 Internet Archive Crawler

Mike Burner’s description of the Internet Archive Crawler [16] was the first paper that focused on the challenges caused by the scale of the Web. The Internet Archive (IA) crawling system was designed to crawl on the order of 100 million URLs. At this scale, it was no longer possible to maintain all the required data in main memory. The solution proposed by the IA paper was to crawl on a site-by-site basis, and to partition the data structures accordingly. The list of URLs to be crawled was implemented as a disk-based queue per Web site. To avoid adding multiple instances of the same URL to the queue, the IA crawler maintained an in-memory Bloom filter [17] of all the site’s URLs discovered so far. The crawl progressed by dequeuing a URL, downloading the associated page, extracting all links, enqueuing freshly discovered onsite links, writing all off-site links to disk, and iterating. Each crawling process crawled 64 sites in parallel, using non-blocking input/output (I/O) and a single thread of control. Occasionally, a batch process would integrate the off-site link information into the various queues. The IA design made it very easy to throttle requests to a given host, thereby addressing politeness concerns, and DNS and robot exclusion lookups for a given Web site were amortized over all the site’s URLs crawled in a single round. However, it is not clear whether the batch process of integrating off-site links into the per-site queues would scale to substantially larger Web crawls.
Brin and Page’s paper outlining the architecture of the first generation Google [18] system contains a short description of their crawler. The original Google crawling system consisted of a single URL server process that maintained the state of the crawl, and around four crawling processes that downloaded pages. Both URL server and crawlers were implemented in Python. The crawling process used asynchronous I/O and would typically perform about 300 downloads in parallel. The peak download rate was about 100 pages per second, with an average size of 6 KB per page. Brin and Page identified social aspects of crawling (e.g., dealing with Web masters’ complaints) as a major challenge in operating a crawling system.

1.9 Mercator Web Crawler

With the Mercator Web crawler, Heydon and Najork presented a “blueprint design” for Web crawlers [19, 20]. Mercator was written in Java, highly scalable, and easily extensible. The first version [19] was non-distributed; a later distributed version [20] partitioned the URL space over the crawlers according to host name, and avoided the potential bottleneck of a centralized URL server. The second Mercator paper gave statistics of a 17-day, four-machine crawl that covered 891 million pages. Mercator was used in a number of Web mining projects [21, 22, 23, 24, 25], and in 2001 replaced the first-generation AltaVista crawler.

1.10 Polybot Web Crawler
Shkapenyuk and Suel’s Polybot Web crawler [26] represents another “blueprint design.” Polybot is a distributed system, consisting of a crawl manager process, multiple downloader processes, and a DNS resolver process. The paper describes scalable data structures for the URL frontier and the “seen-URL” set used to avoid crawling the same URL multiple times; it also discusses techniques for ensuring politeness without slowing down the crawl. Polybot was able to download 120 million pages over 18 days using four machines.

1.11 WebFountain Crawler

The IBM WebFountain crawler [27] represented another industrial strength design. The WebFountain crawler was fully distributed. The three major components were multi-threaded crawling processes (“Ants”), duplicate detection processes responsible for identifying downloaded pages with near-duplicate content, and a central controller process responsible for assigning work to the Ants and for monitoring the overall state of the system. WebFountain featured a very flexible crawl scheduling mechanism that allowed URLs to be prioritized, maintained a politeness policy, and even allowed the policy to be changed on the fly. It was designed from the ground up to support incremental crawling, i.e., the process of recrawling pages regularly based on their historical change rate. The WebFountain crawler was written in C++ and used MPI (Message Passing Interface) to facilitate communication between the various
processes. It was reportedly deployed on a cluster of 48 crawling machines [28].

1.12 UbiCrawler

UbiCrawler [29] is another scalable distributed Web crawler. It uses consistent hashing to partition URLs according to their host component across crawling machines, leading to graceful performance degradation in the event of the failure of a crawling machine. UbiCrawler was able to download about 10 million pages per day using five crawling machines. UbiCrawler has been used for studies of properties of the African Web [30] and to compile several reference collections of Web pages [31].

1.13 IRLbot

Yan et al. described IRLbot [32], a single-process Web crawler that is able to scale to extremely large Web collections without performance degradation. IRLbot features a “seen-URL” data structure that uses only a fixed amount of main memory, and whose performance does not degrade as it grows. The paper describes a crawl that ran over two months and downloaded about 6.4 billion Web pages. In addition, the authors address the issue of crawler traps (Web sites with a large, possibly infinite number of low-utility pages), and propose ways to ameliorate the impact of such sites on the crawling process.

There are a number of open-source crawlers, two of which deserve special mention. Heritrix [33, 34] is the crawler used by the Internet
Archive. It is written in Java and highly componentized, and its design is quite similar to that of Mercator. Heritrix is multithreaded, but not distributed, and as such suitable for conducting moderately sized crawls. The Nutch crawler [35, 36] is written in Java as well. It supports distributed operation and should therefore be suitable for very large crawls; but as of the writing of [36] it has not been scaled beyond 100 million pages.

1.14 Page Rank and HITS

If somebody is surfing the Web for a long time by traversing a link out of each page then different pages will be surfed at different frequencies. The pages having a large number of in-links will be surfed more frequently than the ones having a less number of in-links. The popularity of a page P can be seen as the number of links to P that exist on the whole Web. A Web page P that is linked to by many pages is more popular than the one which is seldom referenced. This citation metric is used for evaluating the impact of published research papers throughout the world.

The back-link approach considers each link the same for finding the importance of a Web page. Brin and Page [18] invented an approach in which the rank or importance of a page (PageRank) is recursively defined in terms of weighted sum of back links to P. The PageRank algorithm simulates the Web as a graph and then used individual node’s visited rate as a popularity measuring score. For a query response all the matching documents are arranged as according to this score. One of the reasons
behind quick response in PageRank is that its calculation is independent from the query.

Bharat et al. [37] made use of connectivity servers to store the pre-crawled portion of the Web to improve upon the speed of crawling. Hyperlink Induced Topic Search (HITS) [38] is an extension to PageRank, it do not need to crawl the Web. HITS works on a search engine. A query to HITS is given to a search engine that retrieves a sub graph of the Web whose nodes match the query terms. The pages having out links or in-links to this graph are also appended to the graph. The extended graph is processed for popular nodes. After processing, the authority scores and hub scores are determined. Authority score for a page is a measure of the references being made by other pages in the sub graph and hub score of a page is a measure for the number of references being made to the other pages in the set. Dean and Henzinger [39] extended the HITS algorithm to Companion algorithm which finds similar pages on the Web using hub based analysis only.

The generic search process makes use of topic-independent crawlers for constructing index and Web page collection for answering the user’s queries. Early implementations of search engine use breadth first crawling and PageRank [18]. Focused crawlers attempt to focus the crawling process towards relevant pages. They tend to keep total number of downloaded Web pages [40, 41, 42] to a minimum, and try to maximize the total number of pages relevant to the domain. The focused crawler is
highly sensitive towards the quality of the initial pages given to start the crawl. These initial pages given to the focused crawler are called the seed pages. Seed pages can be given manually or can also be driven from a set of highly related pages returned by a search engine [43] in reply to the queries related to a specific domain [44, 45, 46]. Good seed is the one which is highly related to the topic or the one which is pointing to many relevant pages. For a topic concerning academic courses available, a good seed page may be the homepage of any university’s or institute’s Website, or any academician’s profile page showing the expertise or research field along with the working institute.

The exponential growth and dynamism of the Web make it difficult to maintain the whole Web index. The focused crawler can be considered as a solution to the exponential growth and huge dynamism problem as it looks only for the pages related to a specific domain and discards the others at the same time which leads only to a small portion for indexing for individual domain. Focused crawlers are guided by rich context like topics and queries etc which helps the crawler to interpret and select the next link to crawl.

The efficacy of a crawler depends on the quality of the heuristics being used for crawling. While crawling, the focused crawler should target only the pages relevant to the topic and should be able to score the pages correctly in order to discard the irrelevant pages. Scoring heuristic used must not be so strict that the related pages could not found by the crawler.
1.15 Fish Search Based Crawler

Kornatzky et al. [47] proposed the fish search based focused Web crawler. They made use of client-based retrieval for documents, which makes use of depth-first search. They compared the search agents exploring the Web to a school of fish in the sea. When relevant information is found the search agent reproduce it and continue to look for more relevant pages on the Web. The search agent is terminated if a path contains a fixed number of irrelevant pages in continuity or there is a lack of bandwidth required by the search agent. The search agent follows more links from relevant pages based upon keyword and regular expression matching. This type of system needs significant amount of network resources and to avoid this different caching strategies are proposed by researchers.

The fish-search take a URL and search query as input and builds a priority list of URLs to be crawled in future. At each stage one URL is popped and processed. The processed page is then analyzed to decide whether it is relevant to the query or not. A score of 1 or 0 given to the page decides whether to pursue the page in future for crawl or not. When a page is found relevant it is processed for the links contained within it and assigned a depth value. If parent is related then the depth is set to be some predefined value otherwise the depth is set to be one less than that of the parent. When depth becomes zero the link has to be discarded and none of its children are going to be inserted into the crawler queue. Links having
score greater than 0 are inserted as according to the following three rules into the crawler queue:

- **a.** The first $k \times \text{width}$, where $k$ is a constant greater than 0, children of a relevant page are inserted at the head of the list.
- **b.** The width children of irrelevant page are inserted to the list after the last child of the relevant node.
- **c.** All remaining children are inserted at the end of the list.

The fish search assigns a binary value for denoting relatedness or non-relatedness, which seems to be the first drawback for the process. Secondly the Web pruning is performed on the basis of width which is an arbitrary assumption.

1.16 Shark Search Based Crawler

Hersovici et al. [48] proposed an improved version, Shark Search, of fish search by replacing the binary decision making with the fuzzy decisions by applying the Vector Space Model. The Shark Search makes use of propagation of a relevant parent score to its children and at the same time decays the score of an irrelevant page by a constant factor. Another improvement consists of the calculation of score of the children not only by propagating ancestral relevance but also by making use of the meta information contained in the links of the documents. Popularity scoring mechanism updates the metadata justifying the use of a page as access point for the number of related pages. Popularity score along with
relevance score is used as an evaluation metric. These two metrics help the crawler to focus around the topic only and avoiding exploration of other non relevant portion of the Web.

1.17 Document Object Model Based Crawler

Pant, Gautam et al. [49] presented a focused crawler based upon Document Object Model (DOM). It determined the priority of the out links of a Web page from the information contained within the tag tree representation of the Web page. Aggregation nodes were made by the crawler for each out link of the Web page and possibility that it would lead to a relevant page was determined with the help of textual information in the aggregation node. Each page was represented in the form of a tag tree. Each node that appeared in the path to out link from the root of the tree was considered to be the aggregation node and the text appearing to the aggregation node sub tree was considered to be the context of the out link.

Medelyan, O et al. [50] proposed a language focused crawling that makes use of a language classifier that determines whether to pursue a page for future relevant results or not. Two steps were used to construct language based corpora. In the first step, a training set of documents satisfying the language and topic requirements is generated to extract the most distinguishing $n$-grams i.e. $n$-grams having highest $tf-idf$ values. Results returned by the search engine in response to these $n$-grams are treated as seed pages for the focused crawler. A classifier is used in the second phase of the crawler. The domain model is created after
training the classifier. The relevancy of a page is judged by its belongingness to the domain and to the domain model.

Aggarwal et al. [51] proposed the concept of intelligent crawling that combines content of the page, information from the URL, children pages and metrics regarding relevant or not relevant pages for calculating priorities to the pages. This leads to a focused crawling approach that does not need direct user training and learns to crawl.

The behavior of paths reaching to the related pages is considered as one of the major criteria in focused crawling. Bergmark et al. [52] proposed tunneling as an improvement parameter in best-first focused crawler. Because the relevant pages are not always going to be driven from the relevant pages only, but they could also be reached by traversing some irrelevant pages also. The crawler is not always aimed to minimize the downloaded pages only but also to collect the pages of high quality. They found that the longer path history has some role to play in retrieving the relevant pages in addition to the current page metrics.

1.18 Agent Based Focused Crawlers

InfoSpiders [53] makes use of user agents to enhance the crawling process. The agents work independently and try to cover large number of related documents on the Web to obtain a good coverage of the Web. It retrieve the seed pages from the search result of a search engine. An agent is initiated for every page present in the seeds and further the
Focused Crawlers: State of the Art

links contained within it are analyzed for inclusion into the set of links to be crawled from the similarity score of the text of the link. Neural network is used for this prediction process. This predicted score is used for selecting the next URL to crawl. The weights of the neural network are updated according to the similarity score of the contents of the page.

Best first crawlers [53] works by assigning priority values to the URLs by computing text similarity with domain topics with the help of Vector Space Model [54]. Term frequencies are used by most of the best first crawlers to derive the text and topic similarity score. Due to its simplicity and easiness in implementation it is considered as one of the successful approach to focused crawling. [53] has shown that the best first crawler outperforms the InfoSpiders, shark search and breadth first crawling approaches. They further developed N-Best first crawler as an extension to the general best first crawler. In this approach first n-best pages are selected to pursue for the future crawl instead of just one page in case of simple best first crawl.

Best-first crawlers consider the page’s content and topic relevance score as the overall document similarity score. Similarity calculations from basic information retrieval techniques like VSM is based upon simple lexical term matching i.e. the overlapping terms are only considered to contribute towards the similarity score. But the absence of any overlapping term does not mean that the documents are not related because two terms can share some semantic meaning like synonyms or
having same meaning. Any crawler based upon the classical technique will fail to retrieve the documents which are lexically different but semantically similar.

McCown and Nelson [55] examined how search results decay over time and built predictive models based on the observed decay rates. Based on their findings, it could take over a year for half of the top 10 results to a popular query to be replaced in Google and Yahoo; for MSN it may take only 2-3 months. Akilandeshwari, Gopalan [56] presented a parallel Web spider model, based on multi-agent system for co-operative information gathering. The system collect Web pages related to the topic, and then employ two agents: master agent and retrieval agents. Bao et al. [57] proposed CoMiner algorithm to conduct a Web-scale mining in a domain independent manner. The CoMiner algorithm consists of the following parts:

a. Given an input entity, extracting a set of comparative candidates and then ranking them according to comparability.

b. Extracting the domains in which the given entity and its competitors play against each other.

c. Identifying and summarizing the competitive evidence that details the competitors' strength.

1.19 Semantics and Learning Based Crawlers
Semantic crawlers can be seen as a solution for the problem of failure of a general focused crawler for retrieving the pages which are lexically different but semantically similar. Semantic crawlers make use of term taxonomies or ontologies to find the semantic similarity amongst the Web pages. Taxonomies or ontologies have terms which are conceptually similar and are connected by IS-A relationship. Some of the topic related ontologies are presented by [58, 59]. The terms having conceptual similarity with the focused topic are retrieved from the taxonomy or ontology and are used to enhance the topic description. After this enhancement document similarity can be determined by applying VSM or Semantic Similarity Retrieval Model (SSRM) [60], or by the method proposed by Corley et al. [61].

The user preferences are learned by the learnable crawler from the training set. The user specifically provide two sets of pages: one set related to the topic and other not related to the topic. The paths leading to the relevant pages are learned. The crawling process categorizes each page into relevant or irrelevant set; a priority value is assigned to it. Some of the initial ideas to learning crawlers make use of Naïve Bayesian classifier for classification of a page in the relevant or irrelevant pages [62] set, some other approaches use Neural Networks, First Order Logic [63], Decision Trees [64] and Support Vector Machines (SVM) [65]. Pant and Srinivasan [66] applied SVM to page content as well as to the link context, and resulting approach outperformed other approaches which uses only page content or link context.
Rennie and McCallum [67] proposed a crawler based upon reinforcement learning [68]. They trained the crawler with some topic related Web pages. A Web page is crawled for a number of times to find the optimized path to reach out to the page. This approach needs the initialization of seed pages from the user which can slow down the crawling process. The approach can further face problems when a specific hierarchy is distributed amongst many numbers of sites.

Diligenti et al. [69] proposed the use of context graphs in focused crawling. Concept of back links is used to reach out to the pages leading to the relevant pages. The pages so collected from back links traversal is used to construct the context graph with different distances. The classifiers were made to classify the pages as according to the distance in the context graph. These classifiers were used by the focused crawler to estimate the priorities of the pages and also to assign priority to the links extracted from these pages.

Yossef et al. [70] extended the digital libraries with the help of topical crawlers which create the collection of documents. The collected pages are processed for lexical and linkage analysis to guide the crawler and to analyze the collection automatically using machine learning. Random walks were used to find the aggregated queries related to the Web pages.

Davison [71] concluded that the probability of similarity of linked pages having similar data is large. The possibility of a child page to become
similar is high if the parent links are closer. Titles, anchor text and description may represent some other part of the relevant pages. A document collection ranked by the human field expert was used for evaluation of different context-based and link based algorithms by Brian et al. [72]. Link-based techniques performed better than the other approaches for the purpose.

A learning focused crawler based upon the anchor text similarity, the parent page similarity, URL words similarity and the surrounding text to predict the relevancy of the unvisited page is proposed by Mejdl et al. [73]. Naive Bayes classification technique is used for judging whether a page belongs to the relevant set or not. The use of arbitrary user-defined predicates is proposed by Aggarwal et al. [74]. Only the pages satisfying the arbitrary predicates are processed by then crawler for generation of relevant pages.

An extension to reinforcement learning for focused crawling approach using neural networks is proposed by Grigoriadis, A. [75]. They represented a page by 500 binary values. The state of a page is defined by Temporal Difference Learning to minimize the state space. The presence of a set of keyword in the page determines relatedness of the page to the topic. Neural network is used at different stages to calculate the similarity values. While training the crawler randomly select pages for a limited number of steps or until it find a relevant page. A step denotes the details of action which accelerates the agent from current state to the next state.
The rewards gained from the state and the features of the state are used to derive the neural network. The neural network is trained to determine a state’s ability to reach out to the successful path. The priorities calculated from the neural network are used by crawler to maintain a list of the links to be followed in future. The state’s value of a children is derived from the value of its parents.

Chakrabarti et al. [76] presented a focused crawling approach that makes use of two classifiers. The Open Directory Project (ODP) is used as the Web taxonomy to classify the visited pages as relevant or irrelevant. The ODP is given to the second classifier to learn to predict whether a page will lead to other relevant page or not.

Bergmark [77] proposed the use of document clustering information as topic descriptors to guide the focused crawler. Huang [78] proposed an intelligent focused crawling method which evaluates the relevance of a page to the given topic by using domain ontology as well as hyperlinks connection to the Web pages related to the domain. They retrieved links and contents from the visited pages. After preprocessing the relevant concepts from the domain ontology, the link structures from the pages are extracted and processed for their numbers. The links prediction score is determined by analyzing the links information. The contents information is used to determine the contents relevance related to e-governance domain ontology. The final score is a combination of the content similarity score and the link prediction score. The larger crawl score pages will be visited
more quickly. This method enhanced the crawling by bootstrapping the domain ontology using machine learning algorithms.

Marc Ehrig, Alexander Maedche [79] proposed an approach for document discovery based on a comprehensive framework for ontology-focused crawling of Web documents. The framework includes the means for using a complex ontology and associated instance elements. It defines several relevance computation strategies and provides an empirical evaluation which has shown promising results. Ismail et al. [80] presented a rule-based focused crawler that used linkage statistics among topics to improve a baseline focused crawler’s coverage.

Liu et al. [81] proposed the concept graph based focused crawling. They made use of Hidden Markov Model (HMM) to predict relatedness of a page to the crawler domain. They proposed three stage process: user data collection, user modeling, and focused crawling. The pages followed by the user for a specific browsing interval were collected in the first phase. The collected pages were clustered. The link structure amongst the clustered pages is used to learn paths which are likely to traverse the relevant pages. HMM is used for the learning process. The priority of links is determined from the possibility of how likely the page is going to lead to a relevant page.

Liu, H. et al. [82, 83] extended the context graph crawling approach with the use of Hidden Markov Model (HMM). In their approach the user marked the pages as relevant or irrelevant while browsing the Web and the
pattern of the marking was recorded to denote the paths leading to the relevant or irrelevant pages. Those patterns were further used to train the crawler.

Pant and Shrinivasan [84] compared different classification schemes. They modeled the crawling process as a parallel best-first search over a graph defined by the Web. The classifiers provided heuristics to the crawler and thus biasing it towards certain portion of the Web graph. Jamali et al. [85] introduced a simple framework for focused crawling using combination of two existing methods, the Link Structure analysis and Content Similarity. M.Yuvrani et al. [86] presented a focused crawler framework that made use of link semantics to retrieve relevant documents and suggested that rule inference mechanisms can be used as a future work to enhance the crawling process.

Pant, Shrinivasan [87] investigated the effects of various definitions of link contexts on the crawling performance. They concluded that a crawler that exploits words both in the immediate vicinity of a hyperlink as well as the entire parent page performs significantly better than a crawler that depends on just one of those cues. Also a crawler that uses the tag tree hierarchy within Web pages provides effective coverage.

Tang et al. [88] addressed the problems of cost, coverage and quality, and built a focused crawler for the mental health topic of depression, which was able to selectively fetch higher quality relevant information. They found that the relevance of unseen pages can be
predicted based on link anchor context, but the quality cannot, and therefore estimated quality of the entire linking page, using a learned IR-style query of weighted single words and word pairs, and used this to predict the quality of its links. The overall crawler priority was determined by the product of link relevance and source quality.

Ching-Chi Hsua, Fan Wub [89] proposed the use of context graphs to select and rank the pages relevant to the domain by using the word distributions of the general and focused topic words. They simulated the proposed crawler and shown that their approach outperformed the breadth first crawler. Ayoub Mohamed et al. [90] compared the standard crawler and focused crawler by taking different parameters and after that deduced the situations where a standard crawler is required and where a focused crawler is required.

Antonio Badia et al. [91] built a focused crawler as part of a larger project. The National Surface Treatment Center (NSTCenter) Website was created with the goal to become a premier forum for Navy officers, independent consultants, researchers and companies offering products and/or services involved in the process of servicing Navy ships. In order to help generate content, they developed a focused Web crawler that searched the Web for information relevant to the NSTCenter. They developed a crawling system that achieves significant precision. Debajyoti et al. [92] proposed a domain specific ontology based crawler that makes use of number of query words in the page to find similarity of a page.
Chen [93] proposed focused crawling based upon combination of classical and learning approaches. The crawler can act as a learning crawler functioning with the help of genetic algorithms or can work as a classical crawler. Web page classification technique based upon the context of the page is proposed by Giuseppe [94]. The information regarding classification of the Web pages is derived from the context of the links referring to it.

Ari Pirkola [95] described negligence of historical results and inability to handle intermediate multilinguity as the main problems for any crawler. Xu et al. [96] presented a general framework of focused Web crawling based on “relational subgroup discovery”. Predicates were used explicitly to represent the relevance clues of those unvisited pages in the crawl frontier, and then first-order classification rules were induced using subgroup discovery technique. The learned relational rules with sufficient support and confidence were used to guide the crawling process afterwards.

An ontology-based approach to learnable focused crawling was proposed by Hai-Tao Zheng et al. [97]. The proposed crawler used a framework based upon ontology. An artificial neural network was derived from the domain ontology and was used for classification of Web pages. The proposed crawler outperformed the breadth-first crawler, the keyword-based crawler approach, and the artificial neural network based focused crawler.
Liu et al. [98] presented a framework based on Maximum Entropy Markov Models (MEMMs) for an enhanced focused Web crawler to take advantage of richer representations of multiple features extracted from Web pages, such as anchor text and the keywords embedded in the link URL, to represent useful context. They treated the focused Web crawling problem as a sequential task and used a combination of content analysis and link structure to capture sequential patterns leading to targets. The experimental results showed that focused crawling using MEMMs are a very competitive crawler in general over Best-First crawling in terms of two metrics precision and maximum average similarity.

A decentralized automata-based learning focused Web crawler was presented by Javad Akbari Torkestani [99]. Learning automata was used to learn the most promising pages which led to the target relevant pages. The proposed crawler adapted its configuration. It made a Web graph of the pages related to the target topic. Fatemeh Ahmadi-Abkenari [100] makes use of link independent click-stream to guide the focused crawler. A weighted architecture for a focused was proposed.

Semi-supervised Ontology-learning-based Focused (SOF) crawler is proposed by Dong and Hussain [101]. The proposed crawler uses domain ontology generation and page information formatting techniques. Ontology learning using semi-supervision and a hybrid Web page classification technique is further augmented with a Support Vector
Machine models. Rodrigo Camposet et al. [102] proposed ontology based focused crawling.

A Web page summarization technique to enhance the focused crawler is proposed by Fahim Mohammadi [103]. Vector Space Model to rate the pages as according to their similarity score is used. The Web page summary is used to train the Neural Network which is then used to summarize the pages. The Web page summaries are used to rank the page using Vector Space Model instead of the contents of the pages. The Neural Network is trained with a specific language and native human reader.

A focused crawling approach based upon heuristic derived from Content Block Partition–Selective Link Context (CBP–SLC) is proposed by Tao Peng, Lu Liu [104]. They used link-context derived from the relevance of content blocks and Web page partition algorithm to improve upon the focused crawling efficiency. A weighted classifier was built by repeatedly applying the SVM algorithm. The SVM algorithm is based upon improved \( tf-idf \) approach. They showed that the proposed crawler outperformed breadth-first crawler, best-first crawler, anchor text based crawler, and link-context crawler.

A supervised Forum Crawler Under Supervision (FoCUS) is proposed by Jingtian Jiang et al. [105]. FoCUS crawls the content of forums which are relevant. The relevant information is contained by the threads of the forums. The forums may have different styles which depend upon the software applications being used to maintain the forums. But all the forums
have specific reaching paths to reach the users from initial pages to threads. Based upon this thought the problem of Web forum crawling is mapped to URL recognition problem. Page type classifiers were used to learn accurate regular expressions from the navigation paths.

Tomasz Kuśmierczyk, Marcin Sydow [106] presented a technique for predicting the content of textual Web documents. It works on the features extracted from the Web pages that link the page. Open Directory Project was used to evaluate the proposed approach.

The Web’s topical locality is used by most of the crawlers to extract the pages relevant to a specific domain to support meaningful Web browsing and searching by Web users. Gautam Pant and Padmini Srinivasan [107] showed that the “status locality” can also be considered as another exhibiting feature of the Web. This comes from the assumption that Web pages tend to link to other pages of similar importance or similarity status. This importance status decays with the increasing distance of the links. Status locality like the topical locality may also be used by the Web crawlers. The pages collected by the crawler may include the pages which are topically similar as well as important also by a similar scale. They made use of the Cobb-Douglas utility function to guide the crawler. They showed that there a trade-off between status and topicality of Web collections.

Vassilis Papavassiliou et al. [108] presented a focused crawler that automatically extracts the topical domain-specific monolingual and
bilingual collection of pages from the Web. Muhammad Faheem, Pierre Senellart [109] proposed Web page archiving based upon the type of the application the Web page is made for. They proposed to make the crawler aware of the type of application it is crawling currently to make it refine the list of URLs to process in future. They made the crawler adaptive to decide whether a page belongs to a Web application or not.

1.20 Other Focused Crawlers

A technique for Web Topic Management is presented by Mukherjea as Web Topic Management System (WTMS) [110]. WTMS allows the user to integrate both the searching and browsing. It uses a crawler to collect information related to a specific topic from the Web. They applied the structural analysis techniques and used the concept of optimal hubs and authorities to filter the information.

Different document clustering techniques were studied by Michael Steinbach et al. [111]. Agglomerative hierarchical and K-means clustering for document collection was compared by them. K-means clustering approach was shown to be outperformed by Hierarchical clustering but is shown to be malicious due to its time complexity. K-means is having a time complexity proportional to the number of documents, but it produced inferior kind of clusters. Cho et al. [112] gives a parallel crawler architecture that can enhance the crawling process by instantiating more than one crawlers in a centrally distributed environment.
Pant et al. [113] build a framework and a number of quality measures to evaluate topic driven crawling algorithms, and proved that a mix of exploration and exploitation is essential for

a. Seeking new relevant pages starting from a known relevant subset.

b. Seeking relevant pages starting a few links away from the relevant subset.

Marina Buzzi [114] gives a scheme to permit a crawler to acquire information about the global state of a Website before the crawling process takes place. It require Web server co-operation in order to collect and publish information on its content. Johnson et al. [115] proved experimentally that a rank function that combines analysis of text and link structure yields effective strategies for focused crawling that performed better than Best First strategy.

Bong and Narayanan [116] proposed a local feature selection measure namely, Categorical Descriptor Term for text categorization. The method explicitly chooses feature set for each category by only selecting set of terms from relevant category. P. Srinivasan et al. [117] presented a general framework to evaluate topical crawlers. They identified a class of measures for fair comparative evaluations of crawlers along some dimensions including generalized notions of precision, recall, and efficiency.
Chang et al. [118] presented a survey of the major Web data extraction approaches and compared them in three dimensions: the task domain, the automation degree, and the techniques used. The criteria of the first dimension explained why an IE system fails to handle some Web sites of particular structures. The criteria of the second dimension classified IE systems based on the techniques used. The criteria of the third dimension measured the degree of automation for IE systems.

Ali Seyfi [119] made use of T-Graph (Treasure Graph) to assign visiting priorities to unvisited URLs. The priority assignment was a two step process. The first step applied a sophisticated custom algorithm to retrieve and analyze the important HTML structural contents like topical boundary, and anchor text of unvisited page, based upon which the topical relatedness of the page was predicted. In the second step T-Graph was used to estimate the similarity score of the unvisited URL.

Swati Mali and B.B. Meshram [120] proposed focused crawler architecture based upon page revisit policy and page selection policy. The structural changes and contents changes are taken into account along with forward link count while predicting the priority of the next URL.

M. Bazarganigilani et al. [121] proposed a genetic based approach for focused crawling that makes use of similarity to decide whether a page is relevant to the topic or not. Genetic algorithm is used to find the best combination for prediction of the similarity between pages and topic. The crawler starts from the seed URLs and for each visited Web page, the
similarity function is used by the classifier to predict whether the page is to be explored further or not. If the page is to be explored then the Web page is inserted into the result set. The links going outside from the relevant page are extracted and inserted in the crawling queue. The decay function is used for each page. A page having less similarity score than a predefined threshold are discarded at that time only.

1.2 Contributions of the Work

The main goal of this thesis is to propose a focused crawler architecture that works in an optimized way to seek out the most relevant pages available on the Web for a specific domain. Its major contributions are:

a. A detailed architecture for a multiphase semantic learning based focused crawler is proposed. The proposed crawler makes use of three major criterions to guide the future crawl: VSM based similarity, WordNet based semantic similarity and Hub score based learning of seed pages for the next phase.

b. To contrast the proposed focused crawler, four variant of other focused crawlers are designed and implemented.

c. Extensive set of experiments involving different focused crawling domains are performed.
Crawling results for the proposed crawler are compared with the crawling results from all the four crawlers for quality page retrieval and page retrieval time.