CHAPTER 2

LITERATURE REVIEW

In this chapter the selected publications are reviewed related to the topics covered in this thesis. In a system like the World Wide Web, users find information by following hypertext links from one document to another. When such a system is small, information could be found easily. But size of the Web that consists of millions of web pages, finding information can be a difficult task. Web Crawlers are tools to assist users in Web navigation.

2.1 Introduction to Hypertext System

A hypertext system consists of collection of documents distributed over the web related by links occurring in the text of the web document. When a hypertext document is read, the reader can choose to follow the links embedded in the documents to other linked documents. In [2], the authors proposed memex which allow its users to access a vast collection of online documents and facilitated the user’s navigation among web documents by means of trails. The trails are links among documents created by the users of the memex system. In [5], the authors enhanced the above concept into modern notion of hypertext. End users could enhance the expressive power of texts by adding links among the text. End users browse the documents and follow hyperlinks from one document to another.

In [9], the authors evolved modern hypertext systems where links can bear information. There are two types of End users, reader and author. A reader would construct hypertext trails by creating links as he browsed through documents. Readers of a hypertext navigate among documents, following links. In modern hypertext systems readers do not edit the hypertext as they browse through it. Authors however publish information over the Web. Most hypertext systems were constructed in a single subject domain [6].
In [8][9][10], the authors proposed Hypertext Editing System, the first functioning hypertext systems put to practical use. In [11], the authors proposed a hypertext system that was designed with content and links crossing administrative boundaries. It was distributed in nature. With the advent of World Wide Web [3] documents are located around the world, created and read by millions of authors and readers. With such a large scale of World Wide Web, finding information is difficult and migrating parallel web crawler is designed to solve this problem.

2.1.1 Information Retrieval

Information retrieval is a relatively old field, its algorithms are described in [12] [4]. Much of the early research work done in information retrieval is relevant to Web search engines. The difference between the two lies firstly in Web and its typical collections secondly in Web’s users and its typical searchers. The major problem in IR is to find a relevant selected set of documents from a large collection in response to a query of searcher. The results of a query are a relevant subset of documents from a larger collection. Research in IR focuses on systems for storing documents, viewing documents, query processing, determining the results, user interfaces for querying and user interface for viewing and refining results [13]. The Two metrics used to evaluate the performance of IR systems are Precision and recall [4]. Precision is the fraction of documents returned in response to a query that are relevant to that query. Precision is subjective because what is relevant is completely up to the particular searcher. While recall is the fraction of relevant documents returned against the total number of relevant documents in the entire collection. Recall is subjective because relevance is up to particular researcher; also the set of documents that makes up the collection is not well defined.

In [12], the authors proposed the vector-space model, a model for retrieving documents. The model specifies which set of documents should be retrieved for a particular query and how those documents should be ranked relative to one another. In [14][15], the authors proposed variation to the basic method such as Boolean logic. Early researchers in Information Retrieval provided methods for improving the facilities given to experienced users of IR systems. Later on professionals used to access the IR systems. These professionals use to learn a complex user interface and how to express their
queries. World Wide Web has changed both the profile of the web searcher and conventional IR system. Web searchers are now relatively inexperienced. Boolean queries itself can be very difficult for average web searcher as compared to professional searcher. Professional searcher is more patient while the Web searcher expects an immediate response. Web searcher wants to spend less amount of time as compared with professional searcher. Information Retrieval Systems built to simplify the life of searchers were too complicated [16]. The Text Retrieval Conference includes a track that explores query processing, outcome of query and difficult user interfaces [17]. Systems for querying the Web use fundamental IR work along with new work focused on the Web. Conventional IR research focuses on long queries with comparatively expert searchers [18]. In [22], the authors proposed method for increasing the performance of short queries. In [23], the authors proposed constraints for increasing the precision of 5-to-10-word queries.

2.1.2 Combining Information Retrieval and Hypertext

In [19][20], the authors proposed combining hypertext with traditional information retrieval system. In this method hypertext is navigated by using both traditional hypertext navigation and full-text indexing. In [21][6], the authors proposed IRIS an information retrieval engine on hypertext model. In [24], the authors proposed a combination of text analysis and link analysis to extend the results to set of documents. In [25][26], the authors proposed model that sort document by criteria that indicate their quality. In [25][27], the authors proposed method that uses link information to infer relevance is related to the practice of citation indexing most commonly used in the scientific community. In [28], the authors proposed method for using Web structure information to find documents related to a specific document. In [29], the authors proposed method for finding and eliminating duplicate documents.

2.1.3 Internet Resource Discovery

The Web is an example of distributed publishing system. The two models based on the way they perform queries are distributed model and centralized model. In Distributed model: the query is originated at central location which is then passed on to few remote locations for processing. The remote locations then send back their result to central location. At central location the result is merged and is presented to the searcher. While
in Centralized model: the complete index is maintained at the central location by one system. The searcher’s query is then executed against the single central index. The advantage of using distributed model is the minimized complexity of searching each database as the collection of indices is distributed among many databases. Examples of the distributed model are Netfind [31], WAIS [30], and Harvest [32]. In [31], the authors proposed Netfind, a system that applied a distributed search strategy to the problem of finding information about people on the Web. WAIS used a vector-space retrieval model over a distributed collection of indexes. WAIS suffered with lack of content.

In [32], the authors proposed the idea of a Digest to send queries only to those servers that have answers to a query. Each server in the system creates a digest, an excerpt of the content available at that site, and then the digest is sent to a central location. Examples of the centralized model are Archie [33], Veronica [34] and all Web search engines. Archie [33] indexed a collection of documents (FTP files) that were already being heavily used. Veronica [34] provided for searches of the Gopher system. Participating Gopher sites had their content downloaded and indexed at a central location.

2.1.4 Web Search Engines

Web made the job of publishing and browsing documents easy. It also eased the way to follow links [3]. The technique used for finding resources on the Web is the WWW Virtual Library. The Virtual Library was organized into a hierarchy. Since it was created by humans its quality was not extraordinary [35]. Later on in 1993, three searchable indexes became popular: Jumpstation [36], the WWW Worm [37], and the RBSE index [38]. Each provided a searchable interface to an automatically constructed index of Web pages. Examples of searching sites were: Lycos [39], InfoSeek [40], Yahoo [41], AltaVista [42] and Excite [43].

2.1.5 Meta-Searching

The Meta search queries multiple remote search engines to satisfy a single searcher’s request. Examples are SavvySearch [44] and MetaCrawler [45]. These Meta engines use servers that have overlapping content. Meta-search is desirable because each search engine has some unique documents in its index. Also, each search engine updates
documents with different frequencies hence some search engine will have up to date web documents. Meta search engines can produce better results. The drawback of meta-search engines is a slowdown in performance. Examples of search engines are AltaVista [42], Fast [46] and Google [47].

2.2 Distributed Systems

A distributed system can scale by adding components, services can be replicated or distributed in such a way that the service is always available and the components of the system can be chosen to minimize the cost of the whole. Three aspects of distributed systems research are the overall architecture of distributed systems, load balancing and resource allocation.

2.2.1 Architecture of Distributed Systems

Research in distributed system focuses on providing platforms that are available, reliable and consistent. Early distributed system models are Locus [48], V Distributed System [49] and Eden [50]. The recent distributed systems are Amoeba [51] and Emerald [52]. The authors of the Coign system assert that their system can make the kinds of optimizations usually made by programmers [53]. Searches are run in parallel on all servers at once and are merged, ranked, and presented to the searcher [54]. Ninja provides a platform for building scalable Web services by using many easy-to-configure servers. Ninja’s goal is to minimize the administrative overhead associated with running a large cluster of systems [55].

2.2.2 Load Balancing

Load balancing is applied in distributed system at the node and in parallel systems at the processor level. In [56], the authors proposed different algorithms for distributing work and accurate models for assigning work. In [57], the authors studied load balancing in Amoeba and concluded that centralized decision making outperforms distributed algorithms. Domain Name System (DNS) has been used to balance requests among several servers by having name servers reply with a list of addresses for a particular name [58]. In [59][60][61], the authors proposed method for tracking the load of the constituent systems.
2.2.3 Resource Location

In [62][63], the authors proposed DNS-based solutions that model the communication cost between the client and server, and return server addresses accordingly. These systems solve the availability problem: the name servers can give out only addresses of servers that are known to be responsive.

2.3 Web Characterization

There are very less number of pages of importance that are hidden in large number of unimportant pages. In web characterization the main problem is how to obtain a good sample. The pages with little or less meaningful information content should be excluded. The estimation of importance of web page is necessary [74].

2.3.1 Methods for Sampling

There are two methods for sampling Web pages: vertical sampling and horizontal sampling. Vertical sampling involves collecting web pages based on domain names. Vertical sampling can be done at various levels of the hierarchical structure. When vertical sampling is done at highest-level it selects countries such as .in, .it, .au etc. When vertical sampling is performed at second level, it collects pages produced by members of the same institution/organization (e.g. integraluniversity.ac.in).

Horizontal sampling involves collection of web pages based on criteria of selection that is not based on domain names. There are two methods for collection of data; firstly by using a log of the transactions in the proxy of a large organization and secondly by the use of Web crawler. While using a proxy it is easy to find the pages that are popular, but the period of revisit is not possible to control because it depends on users; while using a web crawler the pages popularity has to be estimated but the revisit period can be tuned[74][75].

2.3.2 Web Dynamics

There are two established fields of Web dynamics; firstly study of the Web growth and secondly the study of the document updates [76]. The study of growth of the number of pages per web site is given by [77]. The study of the document updates; i.e the change of the Web is given by [78]. When studying document updates, for each page and each visit, the information available is access time-stamp of the page, last-modified time-
2.3.2.1 Estimating Freshness and Age

In [80], the authors considered that changes to a page occur at independent time intervals. Web page changes are periodic in nature as updates occur during office timings. Server usually gives the last modified time of the page. If the server does not give the last-modified time of the page, one can check for modifications by comparing the downloaded copies at two different times.

2.3.2.2 Characterization of Web Page Changes

The different time-related metrics for a Web page are:

- Age of the page is given by: time stamp of visited page – time stamp of page modified.
- Lifespan of the page is given by: time stamp of page deleted– time stamp of page created.
- Number of changes during the lifespan of the page.
- Average change interval: lifespan of the page /number of changes during the life span.

Once an estimation of the above values has been obtained, following metrics for the sample are calculated:

- Distribution of change intervals.
- Average lifespan of pages.
- Median lifespan of pages.

Research is carried on lifespan of pages, as the researchers focus on the availability of Web content and researchers also focus on the rate of change of web pages as the rate of change produces a good re-visiting order.

2.3.3 Link Structure

The graph representing the connections between Web pages has a scale-free topology and a macroscopic structure that are different from the properties of a random graph. A
Web crawler designer must be aware of scale free networks and macroscopic characteristics.

2.3.3.1 Scale-Free Networks

In Scale-free networks there are uneven distributions of link as studied in [81]. It is characterized by a few highly-linked nodes that can act as “hubs” connecting several nodes to the network. Examples of scale-free networks related to the Internet are: Number of links on Web pages, email exchanges and user participation in interest groups and communities. There are Web sites with large numbers of links, which are benefited from a good placement in search engines. In [82], the authors discussed that each Web page creates link to existing Web page. A page with many in-links will attract more in-links than a regular page. In [83], the authors proposed a model in which new node select a random existing node and some links are copied from existing node to new node.

2.3.3.2 Macroscopic Structure

In [84], the authors focused on 200 million web pages from search engine AltaVista and connectivity of these web pages. A Web graph is a single large connected graph. All the pages in this connected graph are reachable from other pages along directed links. Larger strongly connected component is called MAIN. Starting from MAIN if forward links are followed then the component of the graph found are called OUT. If backwards links are followed then the component of the graph found are called IN. The pages that are part of the graph but are not in MAIN, IN and OUT are in another component called TENTACLES. There is a portion of Web sites which are disconnected from the Web graph is called ISLAND component of the web graph.

2.3.4 User Sessions on the Web

User sessions on the Web are characterized by [85] as model of random surfers. The most important source for data about the browsing activity of users is captured in the access log files of Web servers. Several log file analysis software are available such as [86][87][88][89]. Researchers try to infer rules in user browsing history, such as users that visit page A also visit page B. Log file analysis has a number of limitations such as caching and proxies, as given by [90]. Caching implies that re-visiting is a common phenomenon that is not always recorded and can account for about 50% of the activity...
of users [91]. Proxies mean that users can be accessing a Web page from the same IP address. Data preparation is required to process log file data [92][93][94]. An aspect of data preparation is to separate automated/robot sessions from user sessions as studied by [95].

2.4 Indexing and Querying Web Pages

Searching process of the web can be divided into two main parts: firstly on-line process and secondly the off-line process. The on-line process executes every single time whenever the user query is executed, it consist of searching and ranking. The off-line process executes in periodic manner by the search engine itself. It consists of crawling, collecting and indexing. Text normalization operations [96] used for extracting keywords are tokenization, stemming and stopword removal. The process of tokenization consists of partitioning the stream of text into words. In English language the process is to split the text using punctuation and spaces. The process of Stemming derives the morphological root of every single word of the document and is language dependent. For English language rule-based stemming works well while for Spanish language dictionary-based stemming is used. The words that carry very less semantic information are called stopword. In information retrieval stopwords are discarded.

2.4.1 Inverted Index

An inverted index consists of list of occurrences and vocabulary. The vocabulary is a list of sorted keywords and for every term of vocabulary there is a list of all the “places” in which the keyword appearing is stored in the collection. For the query generated the lists are derived from the inverted index and then merged. Practically these queries are fast. The size of the index is determined by the granularity of list of occurrences. If the list of occurrence is small then the index is small. If the list of occurrence is large then the index is large. The list of occurrences is large if the vocabulary grows with the increase in the collection size. The inverted index takes a big space occupied by the actual collection. Then partial indices are built if an inverted index does not fit in main memory for a Web collection. Partial index represents a subset of the collection and are merged to obtain full inverted index.
2.4.2 Distributing Query Load

In the current scenario query response time is required to be very fast and query processing should be done in parallel. The inverted index is distributed among several distinct computers using two techniques: local inverted file and global inverted file as suggested in [97].

When local inverted file is used the document identifiers are divided, but each system gets the full vocabulary. A good load balance is obtained as the query is broadcasted to all computers. This architecture is similar to search engine architecture. When global inverted file is used the vocabulary is partitioned into various parts containing approximately the same amount of occurrences. Each computer system is assigned partial vocabulary and all its occurrences.

For every triggered query, the query is directed to computers holding the query terms and afterwards the results obtained are merged. Load balancing is difficult in this case. The results obtained are shown as 10 or 20 documents per web page. If the user wants it can go past the first or second page of results.

2.4.3 Text-Based Ranking

The vector space model [71] is technique for ranking documents. According to this model document and query is a pair of vectors. Statistical properties from the text are used by Weighting scheme to give certain words more importance when similarity calculations are performed. TF-IDF is used as weighting scheme [72]. To compute the similarity it uses the frequency of the terms in queries and documents.

According to term frequency if a term appears several times in a document, it can be used for describing the contents of that document. IDF (Inverse Document Frequency) implies how frequent a term is in the whole collection. A term that appears in a less number of documents gives more information than a term that appears in many documents.

2.5 Connectivity-Based Ranking

Web links are source of information. In case there are large numbers of pages and there are no methods for establishing the quality of pages, Web links can be used as metric of page quality. Web pages that share a link are likely to be related [98]. Algorithms for
Connectivity-based ranking are discussed in [99]. Connectivity based ranking are of two types: Query-independent ranking and Query-dependent ranking.

2.5.1 Query-Independent Ranking

Query-independent ranking method is also called Hyperlink Vector Voting (HVV). Query-independent ranking assign a fixed score to each page in the collection. The Pagerank [73] is a ranking function used by the Google search engine. A web page with high pagerank is referred by many web pages with high pagerank this is called page rank of a web page. It is a recursive HVV method. The time that the random web user spends at each web page is also the Pagerank of a page. There are millions and millions of Web pages thus the Pagerank can also be manipulated. Another method for ranking pages is based on a Markov chain [100][101]. This method performs better than Pagerank for some information retrieval tasks. Another method for static ranking is the network flow model proposed by [102].

2.5.2 Query-Dependent Ranking

Query-dependent ranking also called topic-sensitive ranking. This method assigns a score to each web page in the collection with respect to context of a specific query. In query-dependent ranking there is neighborhood graph which has pages that are relevant to query. In [103], the authors proposed that a graph is developed by starting with a set of pages containing the query terms; this set is the list of results fetched by a search engine. This set of web pages is called root set. The root set is augmented by the web pages directly pointing to or pointed by pages in the root set. HITS algorithm proposed by authors in [104] considered that relevant web pages can be of two types: authority page or hub page. An authority page has relevant content and a hub page have many links to authority pages. The drawback of HITS algorithm as identified and solved by authors in [105]:

Problem 1: The documents that are in the neighborhood set do not necessarily belong to original topic.

Solution 1: The solution is the analysis of the contents of the documents when executing.
Problem 2: “There are nepotistic, mutually-reinforcing relationships between some hosts”.

Problem 3: “There are many automatically generated links”.

Problem 2 and Problem 3 is solved by following heuristics.

Solution 2 and 3: For the documents on a host to documents in another host if there are k edges. Each edge is given 1/k weight. This provides each document the same degree of influence on the final score, independent of the number of links in document.

In [106], the authors proposed variation of the HITS algorithm designed to avoid “topic drifting”. In this method, for each link, the text in the origin page is compared with text of the destination. If they are similar then link has high weight. This method keeps the pages in the neighborhood closely related. There are two types of link: forward link and backward link. In [107], the authors proposed topic-sensitive Pagerank. In this technique multiple scores for each page are calculated at the time of indexing. Each score represents the importance of a page for each topic. When query is executed the ranking is done using the query to assign weights to the scores of each page.

2.6 Web Crawling Issues

The issues with web crawling are large size and the rate of change of Web. Large number of pages are being added, changed and removed every day. Network speed is also the bottle neck as compared to processing speeds and storage capacities. Web crawler can download a fraction of pages of the World Wide Web pages within a given time frame. So the web crawler is required to prioritize its download. Web crawler retrieves snapshot of the Web [68]. The pages that are crawled last are accurate; the pages that were crawled first have high probability that they were changed. The bandwidth for web crawler is finite and efficient way has to be discovered to retrieve good quality and fresh web pages [108]. A crawler must intelligently select which pages of the queue are required to be next visited. The Web crawler’s behavior is the result of selection policy, revisit policy, politeness policy and parallelization policy

2.6.1 Selection Policy

Selection policy describes which pages to download. Size of WWW is so large that search engines cover fraction of complete web. In [116], the authors demonstrated that
search engine can index only upto 16% of the Web. Web crawler downloads fraction of the WWW, it is necessary that the downloaded content should be relevant. This requires that web pages are prioritized. Designing a good selection policy is challenging task. In [109], the authors proposed policies for crawling scheduling. The stanford.edu domain was used as their data set. In [110], the authors performed web crawling on 328 million pages using breadth-first ordering. In [111], the authors proposed On-line Page Importance Computation in which each web page is provided with initial “cash” which is distributed among the pages it points to. OPIC is similar to a Pagerank. In [112], the authors performed study on 40 million pages from the .it domain and hundred million web pages from the WebBase crawl. In [112], the authors tested breadth-first crawling strategy against random ordering.

Focused crawling proposed by authors in [113] is the importance of a page that can be expressed as a function of the similarity of a page to a given query. In focused crawling problem is to predict the similarity of the text in a given page before downloading the page. In [114], the authors used the complete content of the pages visited to infer the similarity between the query and the pages that will be visited. A focused crawler is used for searching information related to interesting topics from the Internet. The focused crawler does not need to collect all web pages, but selects and retrieves relevant pages only[136] [137] [138]. The crawler must avoid overloading websites or network links as it goes about its business. The web crawler deals with huge volumes of data. Unless it has unlimited computing resources and unlimited time, it must carefully decide what URLs to scan and in what order. The crawler must also decide how frequently to revisit pages it has already seen.

To reduce the crawling traffic, research is going in different areas. The research areas such as network level, crawler level, web server and web crawler coordination are the most popular thrust areas. In crawler level research work, priorities for different types of web pages are given according to their altering rate. Performance of crawling is based on following page changing priorities is incremental crawling [139, 140]. One very interesting approach is to setup Active Routers at the strategic key position in network. These active routers capture the content of web pages from network traffic and keep them for indexing [141]. One proposed solution is to shrink the crawling traffic by coordination between crawler and web server. Special query language and web services
are proposed. Crawler sends the query to Web server and web server will response back
to give details about web pages updates and removals [143]. Other approach is to place
crawler at different geographical positions. These crawlers execute their work locally in
their geographical area [142]. In Mobile crawling, the mobile crawler is placed at the
web server. The web crawler resides on web server, perform their operations on web
server and send updates in return [144]. Researches show that some crawlers do not
obey the moral accepted behavior. Researchers have developed a method to calculate
ethicality of web crawler [145]. Unethical crawlers are causing serious problem for
website. One such trouble is denial of services to real users. It adds extra expenditure to
run the website. Crawler copies copyright property and private information of user
[146].

2.6.2 Re-Visit Policy

Re-Visit Policy inspects the changes in the given web pages. The content of the web is
very dynamic in nature and web crawling is lengthy process that can take months. There
are three events that can happen to web pages: creation, update and deletion [115]. The
event creations signify when a page is created. Page changes can occur at any interval of
time. An update can be either minor or major. A page is deleted if it is removed from
the WWW or its links are removed. In [116], the authors reported that 5.3% of the total
links returned by search engines are deleted pages.

Cost that the web crawlers pay for not detecting an event is to store old copy of a
resource. The cost functions proposed by authors in [117] are freshness and age. The
freshness of the web page implies that the local copy is up to date and accurate. The age
of the web page indicates how outdated or old the local copy is. In [119], the authors
proposed that a crawler must minimize the time web pages remain outdated or old. The
web crawler should aim to keep the freshness high and age of the page low. In [118],
the authors proposed two page revisit policies: Uniform policy and Proportional policy.
In Uniform policy all web pages in the collection are revisited with the same frequency.
While in proportional policy all the web pages that change more frequently are
revisited. The uniform policy is better than proportional policy because when a web
page changes very often, the web crawler will unnecessarily waste time by re-crawling
very often and still the copy of web page will not be fresh [118].
2.6.3 Politeness Policy

In politeness policy the websites cannot be overloaded. In [120], the authors demonstrated that Web robots are useful but following cost is involved in using them:

- Network resources are consumed as robots require high bandwidth.
- Overload servers if the frequency of accesses to a server is very high.

Robot exclusion protocol determines which parts of robots should not access Web servers. In [121], the authors proposed for time interval between two consecutive connections was seconds. In [122], the authors used 10 seconds as access interval, and in [123], the authors used 15 seconds as the default interval for access. In [124], the authors used $10 \times t$ seconds as access interval, if in $t$ seconds a document is downloaded. In [125], the authors used 1 second as interval for access.

2.6.4 Parallelization Policy

A parallelization policy states that how to coordinate distributed Web crawlers. A parallel crawler runs multiple processes in parallel. The major advantage of parallel crawling is that as the size of the Web grows, it becomes imperative to parallelize a crawling process, in order to finish downloading pages in a reasonable amount of time.

In [126], the authors proposed two types of policies Static assignment and dynamic assignment.

Static assignment: In this method the Web is partitioned into sub partitions and each sub partition is assigned to each crawling processes before the actual crawling process may begin.

Dynamic assignment: In this coordination scheme there is a central coordinator that divides the Web into small partitions and each partition is dynamically assigned to a Crawling processes for further downloading.

2.7 Examples of Web Crawlers

The following are web crawler with different components and features:

In [127], the authors proposed RBSE which consists of two programs. First is the spider that maintains a queue in a relational database. Second is the mite which is a ASCII web browser that downloads the web pages from the Internet. In [128], the authors proposed
WebCrawler which is based on lib-WWW to download web pages. In [129], the authors proposed World Wide Web Worm that is a web crawler that build an index of document titles and URLs. In [130], the authors proposed Internet Archive Crawler that is a web crawler designed for archiving snapshots of a portion of the Web at regular interval. Several website are assigned to each process.

In [131], the authors proposed Google Crawler. A URL server transfer list of URLs to crawling processes for downloading. URL server checks that if the URL has been already visited and if that is not the case then the URLs are added to the downloading queue of the URL server.

In [132], the authors introduced CobWeb written on Perl Language that uses a Scheduler which is central in nature and collectors which are distributed in nature. The scheduler uses Breadth first search algorithm and follow the politeness policy so that the web servers are not overloaded.

In [124], the authors proposed Mercator, a modular crawler written in Java language. There are protocol modules and processing modules in Mercator. Protocols modules are responsible for acquiring the Web pages and processing modules are responsible for processing of Web pages. Standard Processing module parses the web pages and extracts URLs while other processing modules index the text of the web pages.

In [108], the authors proposed WebFountain which is a modular, distributed crawler written in C++ programming language. In this crawler there is a controller machine that coordinates with several ant machines. Change rate is calculated for each page downloaded and a non-linear programming technique is used to solve system of equation for maximizing freshness of web page. In [133], the authors proposed PolyBot which is a web crawler written in C++ and Python languages. PolyBot consist of crawl manager, one or more downloaders and DNS resolvers. URLs that are collected, reside in queue on disk and are processed later in batch mode. PolyBot Crawler also follows politeness policy.

In [134], the authors proposed WebRACE, a crawling and caching module that is implemented in Java programming language and is part of system called eRACE. The WebRACE crawler acts as a proxy server. The most important feature of WebRACE is
that it receives starting URLs to crawl continuously while other web crawlers start with seed URLs.

In [135], the authors proposed a distributed crawler called UbiCrawler which is written in Java. It has no central process. It consists of agents identical in nature. Consistent hashing of the host names is used to calculate the assignment function. The advantage of this crawler is that no page is crawled twice. This crawler is fault tolerant and highly scalable.

### 2.8 Web Caching

A Web cache stores Web pages that are expected to be requested again. The objects stored by Web caches are of different sizes. As a result, cache operators and designers track both the overall object hit rate and byte hit rate. Object hit rate is the percentage of requests served from cache and byte hit rate is the percentage of bytes served from cache. Traditional replacement algorithms uses object of fixed size since variable sizes may affect their performance adversely. Web resources that are larger have a higher response times for retrieval, but even for equally-sized resources, the retrieval cost differ because of various reason including distance traveled, network congestion or server load.

A user request may be hit if a response is captured in a browser cache, if it is miss then, it may be passed to a organization wide proxy cache. Response is hit if it captured in organization wide proxy cache, if it is a miss in organization wide proxy cache then, the request may be received by a proxy cache operated by the client's ISP. Response is hit if it is captured in ISP cache; it is a miss if the ISP cache does not contain the requested response. The attempt will be made to contact the origin server. At the origin server, content can be stored in a server-side cache so that the server load can be reduced. An alternative strategy is to use a cooperative caching architecture in which caches communicate with each other using an inter-cache protocol (ICP). In this strategy, on a miss, a cache would inquire a set of peers for the missing object. If there is a hit, the request would be routed to the first responding cache. It there is a miss in all the peer caches, the cache attempts to retrieve the object directly from the origin server.

Web caching has three primary benefits. Firstly, it can reduce network bandwidth, which is beneficial for both clients and servers through more efficient and effective
usage of network resources. Secondly, it can reduce user-perceived delays. The longer it
takes to retrieve content for the user, lower will be value as perceived by the user.
Finally, caching can reduce loads on the origin servers. Potential problems associated
with Web caching are old or out-of-date content, compared to fresh content available on
the origin server. Caching improves the response times for cached responses that are
hits. Cache benefits from requests whose contents are already stored in it. Caching is
hindered by a high rate of change for Web resources.

2.8.1 Web Indexers

A Web indexer takes pages crawled from the Web and generates an inverted index of
those pages for later searching. Google, AltaVista have indexers that perform this
function. However, many early search engines indexes less than the full text of each
web page. Worm (an early search engine) indexed titles and anchor text. Lycos used to
index 20% of the text. Search engines recently index the text and the title of each page.
Intranet search engines may opt for reduced storage and reduced indexes. Indexers
typically include anchor text (text within and/or around a hypertext link) as some of the
terms that represent the page on which they are found. Google indexes anchor text. One
drawback of anchor text is that text might not in fact be related to the target page.

2.8.2 Search Ranking and Community Discovery Systems

Search engines use text analysis to find documents relevant to a query. Web search
engines also incorporate additional factors of user popularity, link popularity and status
calculations. Both status calculations and link popularity depend on the assumption that
page authors direct their readers to pages of interest that are relevant to the topic on the
current page. The approach of link analysis depends on having a set of interconnected
pages which are relevant to the topic and highly connected in order to calculate page
status.

2.8.3 Meta-Search Engines

Meta-search engines are search services that do not search an index of their own, but
instead collect and compile the results of searching using other search engines. These
services present the results they obtained for the client, they rank the results or perform
additional processing. The meta-search engine has the information provided by the
original search engines and the quality of these page descriptors is important for textual ranking or clustering of the pages.

2.8.4 Focused Crawlers

Focused crawlers are Web crawlers that follow links that are relevant to the client's interest. They use the results of a search engine as a starting point or they crawl the Web from their own dataset. Focused crawlers find highly relevant pages using local search starting with other relevant pages. Focused crawlers use the content of the current page or anchor text to determine whether to expand the links on a page or not.

2.9 Crawling Strategies

Search engines maintain central repositories of web pages. Search engine create indexes for the repository. Web crawlers also known as spiders are programs that download pages for search engines. The working of a crawler can be explained as follows:

1. Web pages are downloaded starting with initial set of seed URLs.
2. The web pages are parsed and the links present in the web pages are extracted and then downloaded and are stored in the repository.
3. Links extracted are stored in web crawler's queue and algorithms are applied to prioritize them.
4. The process repeated until no URLs are left in the web crawler’s queue.

Web Crawler is different from viruses or intelligent agents as they do not move over the Internet. Web Crawler resides on a single computer and sends HTTP requests over the Internet to different machines. The main features of a crawler are:

• **Politeness:** Web servers have hidden policies for regulating the web crawler visit rate. Web crawler should consider the politeness policies.

• **Robustness:** Web server creates traps called as spider trap which generate web pages that instruct web crawler to fetch indefinite number of pages. All spider traps are not malicious. They may be result of erroneous website development. So the web crawlers should be intelligent enough to handle spider traps.

• **Distributed:** Web Crawler should have the ability to work in a distributed environment over the Internet.
• **Quality:** It is completely impossible to index the entire WWW pages and large numbers of web pages are of no use for end users, so the web crawler should crawl the relevant pages first.

• **Scalable:** The web crawler architecture should be able to expand the crawl rate by appending machines and network bandwidth in the existing architecture.

• **Freshness:** Web pages are dynamic in nature. The web crawler should revisit the crawled web pages in periodic or incremental manner so that repository is kept updated.

• **Performance and efficiency:** The web crawler must utilize the available processor efficiency, storage space and network bandwidth in best possible manner.

• **Extensible:** Web crawler is designed such that it may handle changes in data format and fetch protocol. The remedy is that the web crawler is designed in a modular fashion.

The best web crawlers should support all the above features but robustness and politeness are essential features and few other features like extensibility are desirable.

As the web crawler cannot visit the whole of World Wide Web due to time constraint it is mandatory for Web Crawler to download important pages first. In [109], the authors proposed metrics for estimation of importance of web pages, which help in deciding the order of important pages that need to be downloaded.

Page similarity is computed in the following manner, document is considered as an \( m \)-dimensional. Significance of word is denoted by \( w_i \).

\[
w_i = (\text{frequency of } i^{th} \text{ term in the document}) \times \text{idf} \text{ of the } i^{th} \text{ word}
\]

where \( \text{idf} \) is the inverse document frequency of the \( i^{th} \) word.

The inverse document frequency is a measure of how much information the word provides, that is, whether the term is common or rare across all documents. It is the logarithmically scaled fraction of the documents that contain the word, obtained by dividing the total number of documents by the number of documents containing the term, and then taking the logarithm of that quotient.

\[
\text{idf}(t, D) = \log \frac{N}{|\{d \in D : t \in d\}|}
\]
with

- \( N \): total number of documents in the corpus \( N = |D| \)
- \( |\{d \in D : t \in d\}| \): number of documents where the term \( t \) appears (i.e., \( tf(i, d) \neq 0 \)). If the term is not in the corpus, this will lead to a division-by-zero.

It is therefore common to adjust the denominator to \( 1 + |\{d \in D : t \in d\}| \).

Back Link Count is based on the principle of citation count. A page \( P \) pointed by more links is important than a page which is pointed by less links. The importance \( I(P) \) is estimated by counting back link counts over the entire crawl. Page rank metric recursively defines the importance of a web page \( P \) as weighted sum of the back links to it. Location Metric defines importance of a page \( P \) depending on location. As .com ending URLs are more useful than URLs ending with other extensions. Also URLs having “home” string is important. URLs with fewer slashes are more useful. These parameters can be evaluated by simply seeing the URLs.

### 2.9.1 Crawling Algorithms

Search engines try to traverse the WWW in minimum amount of time and download the important pages first. The fish algorithms, shark algorithms, breadth first algorithms, depth first search algorithms and best first search algorithms fulfill the above stated requirements. The following sections describe these algorithms in brief:

#### 2.9.1.1 Fish Algorithm

The Fish algorithm [160][161][162] is based on the schools of fish metaphor. According to this principle, Fish travels in the particular direction of food. The number of children is dependent on the supply of food. The fish school separates into different direction in search of food. Fish school that go in direction where there is no food, the fish die of starvation. Fish school also die that enters polluted water. The Fish algorithm works as follows:

1. The list is initially empty. There is a “current” node pointer.
2. The current node is checked for relevant information (keywords, regular expressions). The outgoing links from the current node are extracted.
3. The links are selected in the following manner:
• If the current node is relevant then the links are marked as *child of a relevant node* and if the current node is irrelevant, the links are marked as *child of an irrelevant node*.

• Links pointing to other sites are preferred as they generate better nodes distribution.

• The total number of selected links is not equal to *width*.

For a relevant node, link selected > *width*. For a node that takes large time to retrieve, link selected < *width*. Also, sites with a connection with faster *rate* are preferred over slower connection.

• The links have an associated *depth*. When a link is used and it is not relevant, its subordinate links get lower depth. When the Link has depth 0, the URL’s in the retrieved node are discarded. When node is relevant, the links is issued the depth by the *depth* parameter.

4. The first link to a node is searched which is located on a different site.

5. The algorithm stops when time slot expires or when the content of list is Null.

A limitation of fish-search algorithm is that relevance score are assigned in discrete manner, 0 or 0.5 for irrelevant, 1 for relevant. Here the problem is low differentiation of priority of pages. If huge numbers of pages are of same priority and with limited time to crawl the documents at the tail of the list are not crawled resulting in loss of information. Shark Algorithm overcame this limitation.

2.9.1.2 Shark Algorithm

Shark search algorithm [161] assigns fuzzy score to all documents as compared to binary values that are assigned in fish search algorithm. The vector space model is used for implementation compared to regular expression matching that is used by fish algorithm. Fuzzy relevance score is assigned to child who helps during selection of children of a node that has better score. The inherited score propagates to the descendants of the node and the importance of the grandchildren of a relevant node over an irrelevant node is increased. The inherited score of child node is computed as:

If relevance of current node > 0 (it means current node is relevant)

\[
\text{inherited score of child node} = \delta \times \text{similarity score between query and current}
\]
Document

} // where $\delta$ is decay factor.

Else ( //if current node is not relevant)
{
    inherited score_childnode = $\delta$ * inherited score of current node
}

The relevance score of anchor text is computed as:

{ 
    relevance score_anchortext = similarity score( between query & anchor text)
}

The relevance score of anchor textual context is defined as:

If  (relevance score_anchortext >0)
{
    relevance score_anchortextualcontext =1
}

Else
{ 
    the relevance score_anchortextualcontext = similarity(query & textual context of the anchor)
}

The score of the anchor is defined as Score_anchor = $\beta$ * anchor_text_score +(1- $\beta$) * anchor_textual_context_score, where $\beta$ is constant.

The potential score_childnode is computed as Potential_score of child_node = $\gamma$ * inherited_ score of child_node + (1- $\gamma$) * anchor_score of child_node, where $\gamma$ is predefined constant.
2.9.1.3 Breadth First Search Algorithm

In breadth-first search, step by step crawling is performed over the web pages. The first step of crawl is at first level and the next step of crawl is at second level and so on. The process of web crawling starts at a specific web page and then all web pages are explored that are reachable from the starting page. When all pages at the first step are downloaded, the web crawler downloads the pages at the next step or level. The number of pages increases phenomenally from first step to second step. The process is carried out until no new pages are downloadable. This algorithm downloads the important web pages first, but as the crawl progress the quality of the web pages diminishes. The algorithm of breadth first search is given below:

BFS (seedurls)
{
for each link (seedurls) {
  addqueue(front, link); } // links are added in the front of the queue
while (visited < max_pages) //max_pages is predefined constant
  { link = retrievequeue(front);// link is retrieved from the front of the queue
    page = crawl(link); // web crawling is performed
    addqueue(front, extractlink(page));
    if (#front > maxbuffer)// maxbuffer is predefined constant
      retrievelastlinks(front);
  }
}

Where, addqueue(front, link): link is added at the front of the queue. The queue consists of all the uncrawled URLs.

retrievequeue(front): Link is selected from the front of the queue for downloading the web page.

crawl(link): web page is downloaded from the WWW.
extractlink(page): links are extracted that are present in the downloaded web page.

2.9.1.4 Depth First Search Algorithm

In depth first search algorithm web crawling starts from a specific web page and all web pages reachable from the starting web page are explored. Then one of the web pages is chosen out of explored web pages and it is followed. In this way web crawler keeps on retrieving web pages as deep as possible till no new pages are to be downloaded. Once when no more web pages can be retrieved search backtracks until it can move forward and download new pages. Once the backtracking process is completed and no new pages can be downloaded, the web search jumps to new page and process is repeated again and again until the web crawler is stopped.

2.9.1.5 Best First Search Algorithm

This algorithm is used by focused crawlers. The URLs are visited not in the order they are discovered as in case of breadth first search algorithm. Some heuristics is applied to rank the URLs to be visited for downloading relevant web pages first. The URLs are visited in the direction of relevant web pages and visit of irrelevant pages are avoided. This process is also known as a local search algorithm as the web crawler traverses the search space by visiting neighbors of the previously visited web page.

2.10 Other Related Topics

Several related topics are discussed in this section including monitoring techniques, Site selection, page change frequency, Life span of web page and web page change.

2.10.1 Monitoring Technique

There are two types of crawling schemes: active crawling and passive crawling schemes. In active crawling, a web crawler visits pages of interest periodically to see if they have changed. In passive crawling scheme, a proxy server tracks the fraction of new pages, based on the demand of its end users. Active crawling collects better statistics and it also determine what pages to check/ revisit and how frequently the pages should be revisited. Crawling activity starts with a list of root pages or seed URLs for sites of interest. Pages of interest are periodically revisited. Also predetermined numbers of pages that are reachable from the seed URLs are visited in breadth first manner. This gives us a set of pages whose contents may differ from one visit to
another visit. Pages may leave the set if they are deleted from the set. Pages may enter the set, as new pages are created. This scheme captures new pages.

2.10.2 Site Selection

A modified PageRank metric is used to measure the popularity of a site. The PageRank calculates the popularity of Web pages and it is modified to compute the popularity of web sites. For this a hypergraph is constructed in which the nodes correspond to Web pages and the edges correspond to the links between the pages. The site ending with “.org” and “.gov” are identified as organization and government sites respectively.

2.10.3 Frequency of Page Change

Suppose a page existed in the set for 50 days. If the page changes 5 times in 50 days then the change interval of the page is 10 days. Here the granularity of the change interval is one day, as one change per day is detected. If page changes more than one time a day and then the page remain unchanged for a seven days, then the change interval is much longer than the true value. About 15% of the pages have a change interval (Day < 15% of pages < week) longer than a day and shorter than a week. Large number of pages changes very rapidly. These frequently updated pages are from the commercial domain. 40% of web pages in the commercial domain changes every day. The pages in education and government domain are static in nature. 50% of web pages in education and government domains did not change at all for five months.

Web pages at commercial web sites that are maintained by professionals are updated very frequently to provide information and these web sites attracts more users. Web pages tend to change very rapidly. The actual rates of change of web pages vary dramatically and drastically. Thus a good web crawling strategy is able to effectively and efficiently track all the changes and provide better data than one that is non-sensitive to changing data.

2.10.4 Lifespan of a Web Page

In this section life span of a web page is discussed. For every crawled page, the life span is the number of days the web page was accessible. The visible lifespan is an approximation to the lifespan of a web page conceived by users. When the user starts to look for information on the Internet from a particular Web site, the user starts from its
home page and follows links to reach to a particular page. The user concludes that the web page of intent does not exist if the web page is not reachable within a few links from the home page as the user cannot follow the link infinitely. Large numbers of pages are accessible for longer periods. Life span of about 70% of the pages over all domains is more than one month. Life span of about 50% of the pages in the education and government domain is more than 5 months. The web pages in the commercial domain have short life span and the pages in the education and government domain have long life span.

2.10.5 Web Page Change

Previous sections focused how a Web page evolves over time or how often a web page changes. Here the focus is on how the Web as a whole evolves over time. It takes about 50 days for 50% of the Web to change or to be replaced by new pages. Also different domains change at different rates. About 50% of the commercial domain change in 11 days. 50% of government domain change in four months. Hence the commercial domain is the most dynamic domain. The education and the government domains are the static domain.

2.11 Conclusion

In this chapter, various publications that are relevant to this thesis are studied and explored. The focus of the above discussion is on web characterization, indexing and querying of web pages. In the literature, connectivity based ranking analysis which is an active research topic in the information retrieval community is discussed. Further web crawling issues are discussed. From the literature review it is evident that the design of Web crawler is not represented well in the literature, there are only few publications available. Web crawling research is affected by business secrecy. There is secrecy involved because there are many concerns about search engine spamming. Also ranking functions and crawling methods are usually not published. There is a need to explore the web crawling issues and methods. To achieve the objective of above literature review an effective model is required for efficient web crawling.