CHAPTER 2

WEB INFORMATION RETRIEVAL: A REVIEW

2.1 THE INTERNET

The Internet [29] is often described as ‘a network of networks’ because all the smaller networks are linked together into the one giant network called the Internet. All computers are pretty much equal once connected to the Internet; the only difference will be the speed of the connection which is dependent on the Internet Service Provider and the modem. It uses TCP/IP based communication architecture to interconnect different type of Local Area Networks (LANs) [30].

The Internet by the late 1990s has evolved into a complex environment. Originally it started as a military communication network but now it is routinely used for five types of operations: (i) long-distance transactions (e.g. e-commerce, form filling, entertainment etc.); (ii) interpersonal communication; (iii) data storage; (iv) research (like data mining, cluster analysis etc.); (v) remote data access and downloading.

The Internet can be precisely defined as follows:

- A network of networks, joining many government, university and private computers together and providing an infrastructure for the use of E-mail, bulletin boards, file archives, hypertext documents, databases and other computational resources.
- The vast collection of computer networks which form and act as a single huge network for transport of data and messages across distances which can be from one place to another anywhere in the world.

The figure 2.1 represents the general architecture of Internet. It may be noted that Network 1 is connected to Network 2 through an Internet Service Provider (ISP).
2.2 THE WORLD WIDE WEB (WWW)

The World Wide Web (WWW) is an interlinked collection of documents formatted using Hyper Text Markup Language (HTML). These documents contain hyperlinks to other documents. The links can point to a document on the same machine or to one on the other side of the world. When a user accesses a web page using its Universal Resource Locator (URL), the documents are transferred to the client machine using Hyper Text Transfer Protocol (HTTP). The browser interprets the document and makes it available to the user. The user follows the links in the presented page to access other pages.

The first World Wide Web server was also developed and implemented as NEXTSTEP in 1989. Berners-Lee and his team at CERN paved the way for the future development of the Web by introducing their server and browser, the protocol used for communication between the clients and the server is HTTP, the language used in composing web documents, Hyper Text Markup Language (HTML) and the URL. The figure 2.2 represents the general architecture of the WWW. HTML documents contain hyperlinks to other documents. Web servers and web browsers are communicating client-server computer programs that distribute documents and information, called web data, over the Internet.

![Figure 2.1: General architecture of Internet](Image)
The user accesses the information from the web through the search engine, which is discussed in the next section.

2.3 SEARCH ENGINE

The WWW has more than 3 billion HTML pages and these web pages gain access through search engines only. Search engine [5, 31] is the key to finding specific information on the World Wide Web. Search engine is a program that searches the document for specified set of keywords and returns a list of documents where any or all of the specified keywords were found. Search engine is a general class of programs that enable users to access the documents on the World Wide Web. On the World Wide Web any information resource a computer file or Web pages are accessed using program called Web browser. Web browser accesses these resources using their URLs. Once a resource is accessed following all hyperlinks appearing in that resource access other resources. Search engines are one of the primary ways that Internet users find Web sites. That’s why a web site with good search engine listings may see a dramatic increase in traffic. Search engine works by fetching as many documents as possible and another program called indexer prepares an index on the basis of keywords appearing in the
documents. There are basically different types of search engines. Search engines index millions of web pages containing a comparable number of distinct terms.

The different tasks performed by search engines are:

- They search the Internet based on select pieces of important words.
- They keep an index of terms they find and record of their location.
- They allow users to look for combinations of words found in that index.

There are different challenges in the searching process, which are of dynamic nature of the Web. The data on the Web is semi-structured. It may conform to one schema today but not tomorrow. The next challenge in searching is user language which varies from user to user. Different users may use different terms for the same concept. The other challenges are volume and heterogeneous nature of the Web data. The data on the web do not adhere to a properly defined schema. Searching involves looking for the same data from different or diverse data sources. Although the architectures of different Web search engines may vary, a typical document search engine generally consists of the following four main components: Web crawler, Indexer, Index database, and Query engine, as shown in the figure 2.3.

![Fig. 2.3: General architecture of search engine](image-url)
A Web crawler, also known as a web spider or a web robot, traverses the web to fetch Web pages by following the URLs of Web pages. The Indexer is responsible for parsing the text of each Web page into word tokens and then creating the Index database using all the fetched Web pages. When a user query is received, the Query engine searches the Index database to find the matching Web pages for the query. Although Web search engines like Google, Yahoo, and MSN are widely used by numerous users to find the desired information on the Web, there are still a number of challenges for enhancing their quality like Freshness, Coverage, [32, 33, 34] Quality of Results, Natural Language Query [35, 36] etc.

The search engine traverses the Web to download a variety of data items / documents. The different types of data items are as listed below:

- **Distributed data:** Data is distributed widely in the world. It is located at different sites and platforms. The communication links between computers vary widely. And also, there is no topology of data organization.

- **High percentage of volatile data:** Documents can be added or removed easily in the World Wide Web. Changes to these documents go unnoticed by others.

- **Large data volume:** The growth of data is exponential. It poses scaling issues that are difficult to cope with.

- **Unstructured and redundant data:** The Web is not involving any editorial process. That means data can be false, inaccurate, outdated, or poorly written.

- **Heterogeneous data:** Data on the Web are heterogeneous. They are written in different formats, media types, and natural languages.

- **Dynamic data:** The content of Web document changes dynamically. The content can be changed by a program such as hit counter that keep tracks of number of hits.

The main module responsible for retrieving the Web pages is the crawler, is discussed in the following section.
2.4 CRAWLER

A crawler [7] is a program that downloads and stores Web pages, often for a Web search engine. Roughly, a crawler starts off by placing an initial set of URLs, in a queue, where all URLs to be retrieved are kept and prioritized. From this queue, the crawler gets a URL (in some order), downloads the page, extracts any URLs in the downloaded page, and puts the new URLs in the queue. This process is repeated. Collected pages are later used for other applications, such as a Web search engine or a Web cache. Figure 2.4, shows the principle working of a crawler.

With Web crawling, the general processes that a crawler takes are as follows:
- Check for the next page to download – the system keeps track of pages to download in a “queue”.

![Flowchart](image-url)
• Check to see if the page is “allowed” to be downloaded – checking a “robots exclusion” file and also reading the header of the page to see if any exclusion instructions were provided to this. Some people don’t want their pages archived by search engines.
• Download the whole page.
• Extract all links from the page (additional web site and page addresses) and add those to the queue mentioned above to be downloaded later.
• Extract all words, save them to a database associated with this page, and save the order of the words so that people can search for phrases, not just keywords
• Optionally filter for things like adult content, language type for the page, etc.
• Save the summary of the page and update the “last processed” date for the page so that the system knows when it should re-check the page at a later date.

Web sites also often have restricted areas that crawlers should not crawl. To address these concerns, many Web sites adopted the Robot protocol, which establishes guidelines that crawlers should follow. Over time, the protocol has become the unwritten law of the Internet for Web crawlers. The Robot protocol specifies that Web sites wishing to restrict certain areas or pages from crawling have a file called robots.txt placed at the root of the Web site.

### 2.4.1 ROBOT. TXT PROTOCOL

Robots are programs that traverse many pages in the World Wide Web by recursively retrieving linked pages. These are used by search engines to index the web contents and spammers use them to scan for email addresses. On many occasions these robots have visited WWW servers but they weren’t allowed for variety of reasons. At times these reasons were robot specific like certain robots retrieved the same file repeatedly [37, 38].

The /robot.txt file is used to give instructions about the site to web robots with the help of “The Robot Exclusion Protocol”. A file is created on the server which specifies an access policy for robots and by this robots are excluded from a server. This file should be accessible via HTTP on the local URL “/robot.txt”. It was chosen because it can be easily implemented on any existing web server and a robot can find the access policy with only
single document retrieval. The drawback of such an approach is that only a server administrator can maintain such a test.

A robot wants to visit a web site URL, say http://www.answer.com/welcome.html. Before it does so, it first checks for http://www.answer.com/robot.txt and finds:

The “user agent:* “ means that this section applies to all robots.

The “Disallow:/ ” tells the robot that it should not visit any page on the site. There are two important considerations while using “/ robots.txt”:

- Robot can ignore the /robot.txt especially malware robots, that scan the Web for security vulnerabilities.
- The /robot.txt file is a publicly available file. Anyone can see what section of your server you don’t want robot to use.

The formal and semantics of the “/robot.txt” is as follows:

The file consists of one or more records, separated by one or more blank lines. Each record contains lines of the form “<field>:<optional space><value><optional space>“. The field name is case insensitive. The record starts with one or more user-agent lines, followed by one or more Disallow lines. Unorganized headers are ignored.

2.4.2 DATA STRUCTURES USED IN WEB CRAWLERS

Different data structures involved in crawling process are as follows:

- Repository: Manages and stores a large collection of web pages. It contains full HTML of every Web page. The web pages are compressed in the repository. Different compression techniques can be used for compressing Web pages, which is a tradeoff between speed and compression ratio. In repository documents are stored one after the other.

- Document index: Keeps information about each document. It is a fixed width ISAM (Index Sequential Access Mode) index, ordered by doc ID including document status, a pointer into the repository, a document checksum, and various statistics.
- **Indexer**: It is a program that “reads” the pages, which are downloaded. Web indexing includes individual Web sites or Web documents to provide a more useful vocabulary for Internet search engines. The indexer also examines the HTML code, which makes up the web and looks for words that are considered important. Words in bold, italics or header tags are given more priority. Different indexing methods are available.

- **Hit list**: Store the list of occurrences of a particular word in a particular document including position, font and capitalization information. There are two types of hits fancy and plain hits. Fancy hits consider hits occurring in a URL, title, anchor text, or Meta tag. Plain hits include everything else. A plain hit consists of a capitalization bit, font size, and 12 bits of word position in a document. A fancy hit consists of a capitalization bit, the font size set to 7 to indicate it is a fancy hit, 4 bits to encode the type of fancy hit, and 8 bits of position.

### 2.4.3 BEHAVIOR OF WEB CRAWLER

The behavior of a Web crawler is the outcome of a combination of policies

- a *selection policy* that states which pages to download,
- a *re-visit policy* that states when to check for changes to the pages,
- a *politeness policy* that states how to avoid overloading Web sites, and
- a *parallelization policy* that states how to coordinate distributed Web crawlers.

There are different types of Web crawlers available in literature. Few important types of web crawlers are discussed here.

### 2.4.4 TYPES OF WEB CRAWLERS

Different types of web crawlers are available depending upon how the Web pages are crawled and how successive Web pages are retrieved for accessing next pages. Some of which are as follows:

#### A. Simple Crawler

Developed by Brain Pinkerton in 1994 [39], is a single process information crawler which was initially made for his desktop. Later it was extended on to the internet. It had a
simple structure, and hence had limitations such of being slow and having limited efficiency.

B. Incremental Web Crawler

An incremental crawler [40], shown in figure 2.5, updates existing set of downloaded pages instead of restarting the crawl from scratch each time. This involves some way of determining whether a page has changed since the last time it was crawled.

Fig. 2.5: Architecture of Incremental Crawler

A crawler, which will continually crawls the entire web, based on some set of crawling cycles. An adaptive model, uses data from previous cycles to decide which pages should be checked for updates. Thus high freshness and results in low peak load are achieved. The Crawl Module crawls a page and saves/updates the page in the Collection, based on the request from the Update Module. Also, the Crawl Module extracts all links/URLs in the crawled page and forwards the URLs to All Urls. Instance of the Crawl Module may run in parallel, depending on how fast we need to crawl pages. Separating the update decision (Update Module) from the refinement decision (Ranking Module) is crucial for performance reasons.
However, it may take a while to select/deselect pages for Collection, because computing the importance of pages is often expensive. For instance, when the crawler computes Page Rank, it needs to scan through the Collection multiple times, even if the link structure has changed little. Clearly, the crawler cannot re-compute the importance of pages for every page crawled, when it needs to run at 40 pages/second. By separating the refinement decision from the update decision, the Update Module can focus on updating pages at high speed, while the Ranking Module carefully refines the Collection.

C. Hidden Web Crawler

A lot of data on the web actually resides in the database and it can only be retrieved by posting appropriate queries or by filling out forms on the web. Recently interest has been focused on access of this kind of data called “deep web” or “hidden web” [41]. In recent days crawlers crawl only Publicly Indexable Web (PIW) i.e. set of pages which are accessible through hyperlinks ignoring search pages and forms which require authorization or prior registration. In reality they may ignore huge amount of high quality data, which is hidden behind search forms. Recent studies have shown that large amount of data is hidden behind these search forms in searchable structured and unstructured databases. A crawler HiWE (Hidden web crawler) has been developed which crawls this high quality data. Page(s) in hidden web are dynamically generated in response to the queries submitted via search forms. Crawling the hidden web is highly challenging task because of scale and the need for crawlers to handle search interfaces designed primarily for human being.

D. Focused Crawler

The concept of focused crawler (also referred as topical crawler) was introduced by Manczer in 1998 and further refined by Soumen, Chakrabarti et al in 1999 [6]. Figure 2.6 shows the block diagram of focused crawler. It has three main components: a classifier which makes relevance judgments on pages crawled to decide on link expansion, a distiller which determines a measure of centrality of crawled pages to determine visit priorities, and a crawler with dynamically reconfigurable priority controls which is governed by the classifier and distiller. The focused crawler aims at providing a simpler alternative for overcoming the issue that immediate pages that are lowly ranked related to the topic at hand. The idea is to
recursively execute an exhaustive search up to a given depth, starting from the “relatives” of a highly ranked page.

---

**E. Parallel Crawler**

As the size of the Web grows, it becomes more difficult to retrieve the whole or a significant portion of the Web using a single process. Therefore, many search engines often run multiple processes in parallel to perform the above task, so that download rate is maximized. This type of crawler is known as a parallel crawler [42, 43]. A parallel crawler consists of multiple crawling processes, which we refer to as C-proc. Each C-proc performs the basic tasks that a single-process crawler does. It downloads pages from the Web, stores them locally, extracts URLs from them and follows links. Depending on how the C-proc split the download task, some of the extracted links may be sent to other C-proc. The C-proc performing these tasks may be distributed either on the same local
network or at geographically distant locations [44]. Figure 2.7 represents the general architecture of parallel crawler.

![General Architecture of Parallel Crawler](image)

**Figure 2.7: General Architecture of Parallel Crawler**

**PARCHYD** [45], proposed Augmented Hypertext Documents, wherein if the links contained within a document become available to the crawler before a crawler starts downloading the documents itself, then downloading of its linked documents can be carried out by other instances of the crawler in parallel. But in the current scenario, the links become available to the crawler only after a document has been downloaded. Therefore this is a bottleneck at the document level for parallel crawling point of view. Therefore it is proposed that meta-information in the form Table Of Links (TOL) consisting of the links contained in a document be provided and stored external to the document in the form of a file with the same name as document but with different extension [46]. This one time extraction of TOL can be done at the time of creation of the document.

At the first state, the document retrieval system has been divided into two parts: the crawling system and the hypertext (augmented) document to be downloaded by the crawling process. Once the TOL of a document becomes available to the crawler, the
linked documents, housed on external sites, can be downloaded in parallel by the other agents of the crawler. Moreover the overlapped downloading of the main documents along with its linked documents on the same site also becomes possible.

At the second stage, the downloading system has been divided into two parts: Mapping Process and Crawling Process. The Mapping process resolves IP addresses for a URL and Crawling Process downloads and processes documents.

F. Distributed Web Crawler

This crawler [47] runs on network of workstations. Indexing the web is a very challenging task due to growing and dynamic nature of the web. As the size of web is growing it becomes mandatory to parallelize the process of crawling. Individual URLs to multiple crawlers, which download web pages in parallel, the crawlers then send the downloaded pages to a central indexer on which links are extracted and sent via the URL server to the crawlers. This distributed nature of crawling process reduces the hardware requirements and increases the overall download speed along with reliability. The general architecture of the distributed crawler is represented in figure 2.8.

At any time, each URL should be assigned to a specific agent, which is solely responsible for it. For any given URL, the knowledge of its responsible agent should be locally available. In other words, every agent should have the capability to compute the identifier of the agent responsible for a URL, without communicating. The distribution of URLs should be balanced, that is, each agent should be responsible for approximately the same number of URLs.

- **Scalability**: The number of pages crawled per second per agent should be (almost) independent of the number of agents. In other words, we expect the throughput to grow linearly with the number of agents.
- **No host overload**: A parallel crawler should not try to fetch more than one page at a time from a given host.
- **Fault tolerance**: A distributed crawler should continue to work under crash faults, that is, when some agents abruptly die. No behavior can be assumed in the presence of this kind of crash, except that the faulty agent stops communicating; in particular, one cannot prescribe any action to a crashing agent, or recover its state afterwards (note that this is radically different from milder assumptions, as for instance saying that the state of
a faulty agent can be recovered. In the latter case, one can try to “mend” the crawler’s global state by analyzing the state of the crashed agent). When an agent crashes, the remaining agents should continue to satisfy the “Balanced locally computable assignment” requirement: this means, in particular, the URLs will have to be redistributed.

Figure 2.8: General Architecture of Distributed Web Crawler

G. Mercator: a scalable and extensible web crawler

Designing a scalable web crawler [48] comparable to the ones used by the major search engines is a complex endeavor as shown in figure 2.9. Mercator, a scalable, extensible web crawler is programmed entirely in Java. Mercator’s main support is for extensibility and customizability. By scalable, we mean that Mercator is designed to scale up to the entire web, and has been used to fetch tens of millions of web documents. We achieve scalability by implementing our data structures so that they use a bounded amount of memory, regardless of the size of the crawl. Hence, the vast majority of our data structures are stored on disk, and small parts of them are stored in memory for efficiency. By extensible, we mean that Mercator is designed in a modular way, with the expectation that third parties will add new functionality. In practice, it has been used to create a
snapshot of the web pages on our corporate intranet, to collect a variety of statistics about the web, and to perform a series of random walks of the web. There are many statistics that might be of interest, such as the size and the evolution of the URL space, the distribution of web servers over top-level domains, the lifetime and change rate of documents, and so on. However, it is hard to know a priori exactly which statistics are interesting, and topics of interest may change over time. Mercator makes it easy to collect new statistics—or more generally, to be configured for different crawling tasks—by allowing users to provide their own modules for processing downloaded documents. For example, when we designed Mercator, we did not anticipate the possibility of using it for the random walk application cited above. Despite the differences between random walking and traditional crawling, we were able to reconfigure Mercator as a random walker without modifying the crawler’s core, merely by plugging in modules totaling 360 lines of Java source code.

Figure 2.9: General Architecture of Mercator
H. Adaptive Crawler

Adaptive crawler [49] is classified as an incremental type of crawler which will continually crawls the entire web, based on some set of crawling cycles. The adaptive model used would use data from previous cycles to decide which pages should be checked for updates. Adaptive Crawling can also be viewed as an extension of focused crawling technology. It has the basic concept of doing focus crawling with additional adaptive crawling ability. Since the web is changing dynamically, adaptive crawler is designed to crawl the web more dynamically, by additionally taking into consideration more important parameters such as freshness, whether pages are obsolete, the way pages change, when pages will change, how often pages change and etc. These parameters will be added into the optimization model for controlling the crawling strategy, and contribute to defining the discrete time period and crawling cycle. Therefore, it is expected that more cycles the adaptive crawler goes in operation, more reliable and refined will the output results.

I. Mobile Web Crawler

K. Hammer and J. Fiedler in [50] initiated the concept of Mobile crawling by proposing the use of mobile agents as the crawling units. According to Mobile crawling, mobile agents are dispatched to remote web servers for local crawling and processing of web documents. After crawling a specific web server, they dispatch themselves either back at the search engine machine, or at the next web crawling servers for future crawling. The crawlers based on this approach are crawling mobile crawlers. The general architecture of mobile web crawler is shown in figure 2.10.

The goal of Mobile Crawling Systems are (a) to minimize network utilization, (b) to keep up with document changes by performing on-site monitoring, (c) to avoid unnecessary overloading of the web servers by employing time realization, and (d) to be upgradable at run-time.

Mobile Crawling System out performs traditional crawling by delegating the crawling task to the mobile agent units and gives the as mentioned advantages:
• Eliminating the enormous processing bottleneck from the search engine site. Traditional crawling requires that additional network be plugged into increase processing power, a method that comes at the great expense and cannot keep up with the current web expansion rate and the document update frequency.
• Reduce the use of the Internet infrastructure and subsequently downgrades the network bottleneck by an order of magnitude. By crawling and processing Web documents at their source, user avoids transmission of data such as HTTP header information, useless HTML comments and Java Script code.
• Employs time realization. Mobile Crawler does not operate on this host during peak time. The administrator can define the permissible time spans for the crawlers to execute.

The driving force behind this approach is the utilization of mobile agents that migrate from the search engine to the web servers, and remain there to crawl and process data locally on remote site which saves lot of network resources.

Figure 2.10: General Architecture of Mobile Web Crawling
The goal of all the crawlers discussed above is to optimize the utilization of the internet resources, in terms of memory, bandwidth, processor throughput etc. Memory utilization can be enhanced by making the crawler decide what is relevant and what is irrelevant and storing the document in the repository in accordance to it. The information retrieval system helps to recover the user specific information from the corpus in a systemic way.

2.5 INFORMATION RETRIEVAL

Information retrieval (IR) is the process of finding information (usually documents) of an unstructured nature (usually text) that satisfies an information need from within a large collection of documents (usually stored on computers) [4]. Initially, information retrieval used to be an activity practiced only by few people engaged in professions like reference librarians and search directions. Now the world has changed a lot, information retrieval is fast becoming the dominant form of information access and millions of people engage in information retrieval every day while they use a web search engine for different activities like surfing their e-mail account, chatting etc. Given the increasing amount of information that is available today, there is a clear need for Information Retrieval (IR) systems that can process the desired information in an efficient and effective way. Efficient processing implies minimizing the amount of time and space required to access information, whereas effective processing means identifying accurately which information is relevant to the user. Traditionally, efficiency and effectiveness are at opposite ends and hence the challenge of to find ways to create a balance between efficient and effective data processing.

An information retrieval process begins when a user enters a query into the system, where a query is a formal statement of information needs. For example search strings issued by users in web search engines are queries. In information retrieval a query may or may not uniquely identify a single object in the collection. Instead, several objects may match the query, perhaps with different degrees of relevancy.
An object is an entity that is represented by information in a database. User queries are matched against the database information. Depending on the application the data objects can be either of the following: text documents, images, audio, mind maps or videos.

Most IR systems compute a numeric score called rank. It helps in determining as to how well each object in the database matches the query. The objects are then after ordered on the bases of the rank. The top ranking objects are then shown to the user. The process is iterated if the user refines his query.

Many different measures for evaluating the performance of information retrieval systems have been proposed. The measures require a collection of documents and a query. The measures discussed below assume a ground truth notion of relevancy: every document is known to be either relevant or non-relevant to a particular query. A brief discussion on important performance measures [51] is given below.

- **Precision**

Precision is the fraction of the documents retrieved that are relevant to the user's information need. It takes all retrieved documents into account. It can also be evaluated at a given cut-off rank, considering only the topmost results returned by the system.

\[
\text{precision} = \frac{\text{no. of relevant documents} \cap \text{no. of retrieved documents}}{\text{no. of retrieved documents}} \quad \text{----- eq. 2.1}
\]

- **Recall**

Recall is the fraction of the documents that are relevant to the query that are successfully retrieved. It can be looked at as the probability that a relevant document is retrieved by the query.

\[
\text{recall} = \frac{\text{no. of relevant documents} \cap \text{no. of retrieved documents}}{\text{no. of relevant documents}} \quad \text{----- eq. 2.2}
\]

- **Fall-out**

The proportion of non-relevant documents that are retrieved, out of all non-relevant documents available. It can be looked at as the probability that a non-relevant document is retrieved by the query.

\[
\text{fall-out} = \frac{\{(\text{non-relevant documents}) \cap \text{retrieved documents}\}}{\text{|retrieved documents|}} \quad \text{----- eq. 2.3}
\]
The different steps involved in the information retrieval system are discussed in the following section.

2.5.1 STEPS INVOLVED IN INFORMATION RETRIEVAL SYSTEM

The information need can be understood as forming a pyramid [52], where only its peak is made visible by users in the form of a conceptual query as shown in figure 2.11. The conceptual query captures the key concepts and the relationships among them.

It is the result of a conceptual analysis that operates on the information need, which may be well or vaguely defined in the user's mind. This analysis can be challenging, because users are faced with the general "vocabulary problem" as they are trying to translate their information need into a conceptual query. This problem refers to the fact that a single word can have more than one meaning, and, conversely, the same concept can be described by surprisingly many different words. [32] have shown that two people use the same main word to describe an object only 10 to 20% of the time. Further, the concepts used to represent the documents can be different from the concepts used by the user. The conceptual query can take the form of a natural language statement, a list of concepts that can have degrees of importance assigned to them, or it can be a statement that coordinates the concepts using Boolean operators. Finally, the conceptual query has to be translated into a query surrogate that can be understood by the retrieval system. Similarly, the meanings of documents need to be represented in the form of text surrogates that can be processed by computer. A typical surrogate can consist of a set of index terms or descriptors. The text surrogate can consist of multiple fields, such as the title, abstract, descriptor fields to capture the meaning of a document at different levels of resolution or focusing on different characteristic aspects of a document. Once the specified query has been executed by IR system, a user is presented with the retrieved document surrogates. Either the user is satisfied by the retrieved information or he will evaluate the retrieved documents and modify the query to initiate a further search. The process of query modification based on user evaluation of the retrieved documents is known as relevance feedback [52]. Information retrieval is an inherently interactive process, and the users can
change direction by modifying the query surrogate, the conceptual query or their understanding of their information need.
2.5.2 MAJOR INFORMATION RETRIEVAL MODELS
The following major models have been developed to retrieve information: the **Boolean** model, the **Statistical** model, which includes the vector space and the probabilistic retrieval model, and the **Linguistic and Knowledge-based** models.

### 2.5.2.1 BOOLEAN MODEL

Boolean model is easy to implement and it is computationally efficient [53] and is considered as the standard model for the current large-scale, operational retrieval systems and many of the major on-line information services use it. It enables users to express structural and conceptual constraints to describe important linguistic features [34]. Users find that synonym specifications (reflected by OR-clauses) and phrases (represented by proximity relations) are useful in the formulation of queries [36, 54]. The Boolean approach possesses a great expressive power and clarity. Boolean retrieval is very effective if a query requires an exhaustive and unambiguous selection. The Boolean method offers a multitude of techniques to broaden or narrow a query. The Boolean approach can be especially effective in the later stages of the search process, because of the clarity and exactness with which relationships between concepts can be represented.

The standard Boolean approach has the following shortcomings: Users find it difficult to construct effective Boolean queries for several reasons [55, 37]. Users are using the natural language terms AND, OR or NOT that have a different meaning when used in a query. Thus, users will make errors when they form a Boolean query, because they resort to their knowledge of English.

### 2.5.2.2 STATISTICAL MODEL

The statistical model is considered the "best match" models [32] as it tries to output the most suitable results for the submitted query. Vector space and probabilistic models are the two major examples of the statistical retrieval approach. Both models use statistical information in the form of term frequencies to determine the relevance of documents with respect to a query. Although they differ in the way they use the term frequencies, both produce as their output a list of documents ranked by their estimated relevance. The statistical retrieval models address some of the problems of Boolean retrieval methods.
A. Vector Space Model

The vector space model [56] represents the documents and queries as vectors in a multidimensional space, whose dimensions are the terms used to build an index to represent the documents. The creation of an index involves lexical scanning to identify the significant terms, where morphological analysis reduces different word forms to common "stems", and the occurrence of those stems is computed. Query and document surrogates are compared by comparing their vectors, using, for example, the cosine similarity measure. In this model, the terms of a query surrogate can be weighted to take into account their importance, and they are computed by using the statistical distributions of the terms in the collection and in the documents. The vector space model can assign a high ranking score to a document that contains only a few of the query terms if these terms occur infrequently in the collection but frequently in the document. The vector space model makes the following assumptions: a) The more similar a document vector is to a query vector, the more likely it is that the document is relevant to that query. b) The words used to define the dimensions of the space are orthogonal or independent.

B. Probabilistic Model

The probabilistic retrieval model is based on the Probability Ranking Principle, which states that an information retrieval system is supposed to rank the documents based on their probability of relevance to the query, given all the evidence available [37]. The principle takes into account that there is uncertainty in the representation of the information need and the documents. There can be a variety of sources of evidence that are used by the probabilistic retrieval methods, and the most common one is the statistical distribution of the terms in both the relevant and non-relevant documents.

The statistical approaches have the following strengths: They provide users with a relevance ranking of the retrieved documents. Hence, they enable users to control the output by setting a relevance threshold or by specifying a certain number of documents to display. Queries can be easier to formulate because users do not have to learn a query language and can use natural language. The uncertainty inherent in the choice of query concepts can be represented.

2.5.2.3 LINGUISTIC AND KNOWLEDGE-BASED APPROACHES
In the simplest form of automatic text retrieval, users enter a string of keywords that are used to search the inverted indexes of the document keywords. This approach retrieves documents based solely on the presence or absence of exact single word strings as specified by the logical representation of the query. Clearly this approach will miss many relevant documents because it does not capture the complete or deep meaning of the user's query. Finally, the linguistic methods [31] have to resolve word ambiguities and/or generate relevant synonyms or quasi-synonyms based on the semantic relationships between words. The development of a sophisticated linguistic retrieval system is difficult and it requires complex knowledge bases of semantic information and retrieval heuristics. Hence these systems often require techniques that are commonly referred to as artificial intelligence or expert systems techniques. The Table 2.1 represents the major characteristics of the different retrieval techniques.

Table 2.1: Characteristics of different major retrieval methods

<table>
<thead>
<tr>
<th>Linguistic level</th>
<th>Boolean Retrieval</th>
<th>Statistical</th>
<th>Linguistic and knowledge base</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lexical</td>
<td>Stop-word list</td>
<td>Stop-word list</td>
<td>Lexicon</td>
</tr>
<tr>
<td>Morphological</td>
<td>Truncation symbol</td>
<td>Stemming</td>
<td>Morphological analysis</td>
</tr>
<tr>
<td>Syntactic</td>
<td>Proximity operator</td>
<td>Statistical phrases</td>
<td>Grammatical phrases</td>
</tr>
<tr>
<td>Semantic</td>
<td>Thesaurus</td>
<td>Cluster of co-occurring words</td>
<td>Network of words/phrases in semantic relationship.</td>
</tr>
</tbody>
</table>

The following section discusses about the Ontologies, which plays an important role in providing meaningful representation to information in the document, thus assisting in the extraction of relevant data. Practically, an ontological commitment is an agreement to use a vocabulary (i.e., ask queries and make assertions) in a way that is consistent (but not complete) with respect to the theory specified by an ontology.

2.6 ONTOLOGY

The term ‘ontology’ is derived from the Greek words ‘onto’, which means being, and “logia”, which means written or spoken discourse. In computer science, ontology formally represents knowledge as a set of concepts within a domain, and the relationships
among those concepts. It can be used to reason about the entities within that domain and may be used to describe the domain. Ontologies [57, 58] are the structural frameworks for organizing information and are used in artificial intelligence, semantic web, systems engineering, software engineering, biomedical informatics and information architecture as a form of knowledge representation about the world. The figure 2.12 represents an example of ontology of beverage, the conceptual framework for organizing the hierarchy of beverages.

![Figure 2.12: Ontology of Beverage](image)

In the context of knowledge sharing, the term ontology means a specification of a conceptualization. That is, ontology is a description (like a formal specification of a program) of the concepts and relationships that can exist for an agent or a community of agents. This definition is consistent with the usage of ontology as set-of-concept-definitions, but more general. What is important is what ontology is for? In the context of knowledge sharing, ontology is a specification used for making ontological commitments. For pragmatic reasons, ontology is selected to write a set of definitions of formal vocabulary, shown in figure 2.13, the specification for the word ‘Spider’.
Although this isn't the only way to specify a conceptualization, it has some nice properties for knowledge sharing among AI software (e.g., semantics independent of reader and context).

Different meanings of the word ‘spider’, can be associated with its characteristics, events, synonyms etc., which can provide meaning full representation for the keywords.

### 2.6.1 ONTOLOGY REPRESENTATION

The architecture of ontology [59] can be represented in the following three forms (as shown in figure 2.14 (a, b, c)):

a) Single Ontology Approach
b) Multiple Ontology Approach
c) Hybrid Ontology Approach (combination of both).

The following paragraphs give a brief overview of the three main ontology architectures.

- **Single Ontology approaches** Single ontology approaches use one global ontology providing a shared vocabulary for the specification of the semantics (fig. 2.14(a)). All information sources are related to the existing single ontology only. A prominent approach of this kind of ontology integration is SIMS [60]. SIMS model of the application domain includes a hierarchical terminological knowledge base with nodes
representing objects, actions, and states. An independent model of each information source must be described for this system by relating the objects of each source to the global domain model. The relationships clarify the semantics of the source objects and help to find semantically corresponding objects.

Figure 2.14(a): Single Ontology Approach

The global ontology can also be a combination of several specialized ontologies. A reason for the combination of several ontologies can be the modularization of a potentially large monolithic ontology. Single ontology approaches can be applied to integration problems where all information sources to be integrated provide nearly the same view on a domain.

- **Multiple Ontologies:** In multiple ontology approaches, each information source is described by its own ontology (fig. 2.14(b)). For example, in OBSERVER [61] the semantics of an information source is described by a separate ontology. In principle, the “source ontology” can be a combination of several other ontologies but it cannot be assumed that the different “source ontologies” share the same vocabulary. At a first glance, the advantage of multiple ontology approaches seems to be that no common and minimal ontology commitment [57] about one global ontology is needed. Each source ontology could be developed without respect to other sources or their ontologies — no common ontology with the agreement of all sources is needed. This ontology architecture can simplify the change, i.e. modifications in one information source or the adding and removing of sources. But in reality the lack of a common vocabulary makes it extremely difficult to compare different source ontologies.
• **Hybrid Approaches:** To overcome the drawbacks of the single or multiple ontology approaches, hybrid approaches were developed (Fig. 2.14(c)). Similar to multiple ontology approaches the semantics of each source is described by its own ontology. But in order to make the source ontologies comparable to each other they are built upon one global shared vocabulary [62][63]. The shared vocabulary contains basic terms (the primitives) of a domain. In order to build complex terms of source ontologies the primitives are combined by some operators. Because each term of a source ontology is based on the primitives, the terms become easier comparable than in multiple ontology approaches. Sometimes the shared vocabulary is also ontology [64]. In hybrid approaches the interesting point is how the local ontologies are described, i.e. how the terms of the source ontology are described by the primitives of the shared vocabulary.

![Diagram of Hybrid Ontology Approach](image-url)
The characteristics of the different ontological approaches are summarized in the table 2.2.

Table 2.2: Comparison of different ontological approaches

<table>
<thead>
<tr>
<th>Characteristics</th>
<th>Single Ontology approach</th>
<th>Multiple Ontology Approaches</th>
<th>Hybrid Ontology Approaches</th>
</tr>
</thead>
<tbody>
<tr>
<td>Implementation efforts</td>
<td>Straight forward</td>
<td>Costly</td>
<td>Reasonable</td>
</tr>
<tr>
<td>Semantic heterogeneity</td>
<td>Similar view domain</td>
<td>Support heterogeneous views</td>
<td>Support heterogeneous views</td>
</tr>
<tr>
<td>Adding and removing of source</td>
<td>Need for some adoption in the global ontology</td>
<td>Providing a new source ontology; relating to other ontologies</td>
<td>Providing a new source ontology</td>
</tr>
<tr>
<td>Comparing of multiple ontologies</td>
<td>-</td>
<td>Difficult because of lack of common vocabulary</td>
<td>Simple because ontologies use common vocabulary</td>
</tr>
</tbody>
</table>

2.7 CONTEXT

Context is a very wide concept and it is worn in several fields. It is difficult to give the exact definition of the term context. Though there have been several attempts to make one. Dey’s [64] defined this word as, “Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or thing that is considered significant to the interaction between a user and an application, including the user and applications themselves”. Context is the where, who, what and when of any living or non-living thing. If a piece of information can be used to characterize the situation of a participant in an interaction or discussion, this information is context of that applicant. A word can be referred differently depending on the background it is brought up. The same word can be interpreted quite different from person to person. People’s environment and understanding is vital to have the same conception of a particular context. Computers however, do not have this ability. It is required that computers should be made more context-aware i.e. computers ability to be aware of and adapt to the context that surrounds them.

In case of information retrieval systems this knowledge can be given to the computer through the ‘ontology’. The ontologies for the keywords can be framed more meaning
fully if the properties of the English words in the documents are identified. Each language has different properties associated with the words in the present in the language dictionary English word properties are discussed in the following section.

2.8 ENGLISH WORDS PROPERTIES

The word present in English language can have different characteristics like they can-Homophones, Synonyms, Heteronyms or Polysemy. The different properties associated with these words are:

- **Homophones**
  Homophones are words that sound the same but are usually spelled differently. They also have different meanings. Words like war (fighting between countries) and wore (past tense of wear) are examples of homophones. The word homophone comes from combining the Greek prefix homo meaning "same" and suffix phone meaning "sound." So the spelling may be different but the sound is the same in homophones.

- **Synonyms**
  As the name implies, synonyms are words that mean the same or have similar meanings in context. Synonyms are used in a variety of situations not only for variety, but to express thoughts or ideas in another, often more emphatic manner. The use of synonyms in a euphemistic manner is especially found in the field of education. 'Failure' has been replaced by 'substandard achievement' so that the word and -- by extension -- the concept of failure has been all but lost.

- **Heteronyms**
  Heteronyms are words that are spelled the same but have different meanings. Words like bank (Coastal area) and bank (Financial institute) are examples of heteronyms. Homonyms, homographs, and heteronyms share some similarities while being distinctly different.

- **Polysemy**
  The concept of metonymy also informs the nature of polysemy, i.e., how the same phonological form (word) has different semantic mappings (meanings). If the two meanings are unrelated, as in the word pen meaning both writing instrument and
enclosure, they are considered homonyms. Within logical polysemies, a large class of mappings may be considered to be a case of metonymic transfer, e.g., chicken for the animal, as well as its meat; crown for the object, as well as the institution.

2.9 CURRENT CONTEXT BASED INFORMATION RETRIEVAL SYSTEMS

Lot of research has been done in the direction of topical web crawling. Some of the recent research papers are discussed in the following section.

A. TOPICAL WEB CRAWLERS: EVALUATING ADAPTIVE ALGORITHMS [65]

Searching the Web is a difficult task. A lot of machine learning work is being applied to one part of this task, namely ranking indexed pages by their estimated relevance with respect to user queries. This is a crucial task because it heavily influences the perceived effectiveness of a search engine. Users often look at only a few top hits, making the precision achieved by the ranking algorithm of paramount importance. Early search engines ranked pages principally based on their lexical similarity to the query. The key strategy was to devise the best weighting algorithm to represent Web pages and queries in a vector space, such that closeness in such a space would be correlated with semantic relevance. More recently the structure of hypertext links has been recognized as a powerful new source of evidence for Web semantics. Many machine learning techniques have been employed for link analysis, so that a page’s linkage to other pages, together with its content, could be used to estimate its relevance. The best known example of such link analysis is the Page-Rank algorithm successfully employed by the Google search engine [5]. Other machine learning techniques to extract meaning from link topology have been based on identifying hub and authority pages via eigen-value analysis [66] and on other graph-theoretical approaches [67, 68, 69]. While purely link-based methods have been found to be effective in some cases, link analysis is most often combined with lexical pre/post-filtering. However, all this research only addresses half of the problem. No matter how sophisticated the ranking algorithm we build, the results can only be as good as the pages indexed by the search engine—a page cannot be retrieved if it has not
been indexed. This brings us to the other aspect of Web searching, namely crawling the Web in search of pages to be indexed. The visible Web with its estimated size between 4 and 10 billion “static” pages as of this writing [70] offers a challenging information retrieval problem. This estimate is more than double the 2 billion pages that the largest search engine, Google, reports to be “searching.” In fact the coverage of the Web by search engines has not improved much over the past few years. Even with increasing hardware and bandwidth resources at their disposal, search engines cannot keep up with the growth of the Web. The retrieval challenge is further compounded by the fact that Web pages also change frequently [71]. For example Cho and Garcia-Molina in their daily crawl of 720,000 pages for a period of about 4 months, found that it takes only about 50 days for 50% of the Web to change. They also found that the rate of change varies across domains. For example it takes only 11 days in the .com domain versus 4 months in the .gov domain for the same extent of change. Thus despite the valiant attempts of search engines to index the whole Web, it is expected that the subspace eluding indexing will continue to grow. Hence the solution offered by search engines: the capacity to answer any query from any user is recognized as being limited. It therefore comes as no surprise that the development of topical crawler algorithms has received significant attention in recent years [72, 73]. Topical crawlers (also known as focused crawlers) respond to the particular information needs expressed by topical queries or interest profiles. These could be the needs of an individual user (query time or online crawlers) or those of a community with shared interests (topical or vertical search engines and portals). Topical crawlers support decentralizing the crawling process, which is a more scalable approach [74, 75]. An additional benefit is that such crawlers can be driven by a rich context (topics, queries, user profiles) within which to interpret pages and select the links to be visited. Starting with the early breadth first and depth-first [76] crawlers defining the beginnings of research on crawlers, we now see a variety of algorithms. There is Shark Search [77], a more aggressive variant of De Bra’s Fish search. There are crawlers whose decisions rely heavily on link based criteria [7]. The present highly creative phase regarding the design of topical crawlers is accompanied by research on the evaluation of such crawlers, a complex problem in and of itself. For example, a challenge
specific to Web crawlers is that the magnitude of retrieval results limits the availability of user-based relevance judgments.

B. ONTOLOGY-SUPPORTED WEB CRAWLER FOR INFORMATION INTEGRATION ON CALL FOR PAPERS [78]

This paper has discussed an ontology-supported OntoCrawlerII for integration on CFP web pages, in which only scholar entered some keywords would the system supported by domain ontology actively provide comparison and verification for those keywords so as to up-rise precision rate and recall rate of webpage searching. This technique has practically been installed in Google and Yahoo search engines and furthermore active searched and filtered out some unduplicated and related CFP web pages and accordingly stored the results into a database to let the backend systems to do advanced processes.

C. ARCHITECTURE OF A CONTEXT-BASED INFORMATION RETRIEVAL SYSTEM [24]

New context-based information retrieval (IR) system where users could indicate the context in which they are seeking information. It uses a novel way of representing context based on the type of information. The context defined in this manner is most intuitive to the user and originates from their information need in a problem-solving situation. Context categories for documents in computing and information technology domain and offers a new methodology for assigning context to documents in a collection are presented. The methodology showed promising results and the inter-assigner consistency quite comparable to the results reported in literature. This new context-based information retrieval system overcomes major limitations of the keyword based IR systems currently in use and provides improved performance.

D. FOCUSED CRAWLING BY LEARNING HMM FROM USER'S TOPIC-SPECIFIC BROWSING [79]

This technique first collect pages that a user visits during a learning session, where the user browses the Web and specifically marks which pages the user is interested in. It examine the semantic content of these pages to construct a concept graph, which is used
to learn the dominant content and link structure leading to target pages using a Hidden Markov Model (HMM). Experiments show that with learned HMM from a user’s browsing, the crawling performs better than Best-First strategy.

E. CONTEXT BASED INDEXING IN SEARCH ENGINES USING ONTOLOGY [80]
This research proposes an indexing structure in which index is built on the basis of context of the document rather than on the terms basis using ontology. The ontology-based collection selection method presented in this paper uses context to describe collections and search engines. The context of the documents being collected by the crawler in the repository is being extracted by the indexer using the context repository, thesaurus and ontology repository and then documents are indexed according to their respective context. The context based index enables retrieval from index on the basis of context rather than keywords. This aids in improving the quality of the retrieved results. A rough estimate of support values for the existing and the proposed system clearly depicts the better performance of the existing system.

F. NOVEL FRAMEWORK FOR CONTEXT BASED DISTRIBUTED FOCUSED CRAWLER [81]
This research provides a novel framework of a context based distributed focused crawler that maintains an index of web documents pertaining to the context of keywords resulting in storage of more related documents. It works on the different contextual interpretations of the keywords entered by the user in the form of query and the keywords present in the web documents. It presents these different contextual meaning of the keywords to the user to select the particular context of search, thus results in more specific web pages to accomplish the requirement. In this paper a framework for CBDFC has been proposed that indexes the web pages with the help of Word Net. Indexing the database is the core of the whole framework and as it is based on the different contextual senses of the query keyword, it focuses the search to the specific context of user interest. It also uses a novel technique to extract the back-links of a URL recursively until it finds the irrelevant links,
enriching the database. It may be noted from the result analysis that the proposed framework results in more relevant and topic specific web documents.

G. CONTEXT-BASED INFORMATION RETRIEVAL FOR IMPROVED INFORMATION QUALITY IN DECISION-MAKING PROCESSES [82]
The definition of information quality tailored to the context of decision-making is provided. The definition of information quality is derived from the semiotic concept of information and results in eight information quality criteria. The research illustrates the technological implementation of quality criteria with a special focus on the relevance of information as one important criterion of information quality. An approach will be described to search automatically for relevant documents using any document that describes the research context to set off the search process instead of a simple search string.

H. DOMAIN EVENT EXTRACTION AND REPRESENTATION WITH DOMAIN ONTOLOGY [83]
With domain ontology, a meaningful index of document indexing, such as the domain events structure is defined. Since the construction of domain ontology is costly, an automatic domain ontology acquisition is preferred. The named entity, identified can be a clue of domain event acquisition from the training corpus. Therefore, it adds the named entity recognition module into the automatic domain ontology acquisition process. Thus, a subject-verb-object-modifier (SVOM) indexing is constructed. Experimental results demonstrate that the automatic event extraction and ontology acquisition can be good resources for text categorization and further information processing.

I. AGENT BASED CONTEXT DRIVEN FOCUSED CRAWLER [84]
The design of a Context Driven Focused Crawler (CDFC) is being proposed that provides the context of the keywords to the user in a flexible and interactive category tree. The agent-based design not only overcomes the complex time-consuming computations of existing focused crawlers but also reduces network traffic significantly. In line with the demands of a focused crawler that the relevant information should be collected and
retrieved by the user in the least amount of time possible, the proposed architecture reduces the search time for a document and the information database on the search engine side becomes more easily manageable.

J. FOCUSED CRAWLER BASED ON DOMAIN ONTOLOGY AND FCA [85]
The research paper proposes, an effective focused web crawling method which based on domain ontology and Formal Concept Analysis (FCA). The method construct a core similarity graph based on Word Net and concept relatedness firstly, and then combining with concept lattice knowledge, a Similarity Concept Context Graph (SCCG) is built. On the basis of SCCG, a focused web crawling method which can measure a page’s expected relevancy to a given topic and determine which URL should be crawled firstly is proposed. Experimental result shows our approach has higher recall rates than the standard breadth-first approach, the approach with Context Graph (CG) and the approach with Relevancy Context Graph (RCG).

2.10 ISSUES AND CHALLENGES OF CONTEXT BASED CRAWLING
A critical look at the available literature indicates that the existing information retrieval systems suffer from following drawbacks:

- The existing crawling system is lacking in the technique of self learning, for identifying the context of the web documents available.
- Crawlers are not able to analyze the exact topical significance of the set of keywords submitted by the user.
- Improvement in decision – making for relevant information retrieval is required.
- Results returned are identical, independent of the interest of the user.
- Lack optimizing the mechanism of priority score computing and enhancing the precision of the topic repository of crawling.
- To organize the keyword and their contexts in a better fashion as storing in a linear fashion makes searching of a document a bit time consuming.
• There is a requirement of such a system which generates the contextual ontology for the different hyponyms, so that the words having the multiple interpretations can be represented in a more meaningful and logical manner to remove the ambiguity for representation both for the search-engine as well as the user.

• Current well developed and understood web crawling and indexing techniques are not directly applicable, for semantic web crawling since they focus almost exclusively on text indexing.

• A semantic web crawler differs from a traditional web crawler as: the format of the source material it is traversing.

• Transformation of World Wide Web into semantic web has been almost impossible due many practical reasons like independent ontologies.

• Crawler treats user search requests without full context and do not focus on the topic that’s why results returned are ambiguous and not satisfy the interest of the user.