Chapter 2

HIDDEN WEB: A REVIEW

2.1 INTERNET AND WWW:

The Internet is a global system of interconnected computer networks that is used to serve billions of users worldwide. It is a network of networks that consists of millions of private, public, academic, business, and government networks. In fact, it has become an important part of our daily routine, with powerful capabilities. This technological marvel was created by the people who had a vision of creating a universal networking environment developed with a view to provide information/multimedia objects at the top of its user.

In later 1950’s, the US military was particularly concerned about the effects of a nuclear attack on their communications infrastructure, if they couldn't communicate, they wouldn't be able to regroup or respond, in the event of the first strike by the Soviet Union. Therefore, the Advanced Research Projects Agency (ARPA) was initiated by US to find safeguards against a space-based missile attack from USSR.

Later on, ARPA established the IPTO in 1962 with a mandate to build a survivable computer network to interconnect the main computers at the Pentagon, Cheyenne Mountain, and SAC HQ [23]. This initiative led to the development of the ARPANET seven years later, and then to the NSFNET and the Internet we know today.

The terms Internet and World Wide Web are often used in our daily words without much difference. However, the Internet and the World Wide Web are not one and the same. Internet is a global data communications system infrastructure that provides connectivity between computers. In contrast, the Web is one of the services communicated via the Internet [24]. It is a collection of interconnected documents and other resources, linked by hyperlinks and URLs. This collection of hyperlinked documents is called as World Wide Web, is a way of accessing information over the medium of the Internet. It is an information-sharing model that is built on top of the Internet.
Before the World Wide Web, the Internet only provided screens full of text. Although it was pretty good for exchanging information, but it was visually very boring. In an attempt to make this more aesthetic, companies like Compuserve and AOL began developing GUIs (or graphical user interfaces). GUIs added a bit of colour and a bit of layout, but were still primitive.

WWW resources are organized to allow users to move easily from one resource to another. Users generally navigate through the WWW using an application known as a WWW browser client. The browser presents formatted text, images, sound, or other objects, such as hyperlinks, in the form of a WWW page on a computer screen. The user can click on a hyperlink with the cursor to navigate to other WWW pages on the same source computer, or server, or on any other WWW server on the network. The WWW links exist across the global Internet to form a large-scale, distributed, multimedia knowledge base consisting of words, phrases, images, or other information. Smaller-scale implementations may occur on enterprise internets [24, 25]. WWW pages are formatted using Hypertext Markup Language (HTML), and information is transferred among computers on the WWW using a set of rules known as Hypertext Transfer Protocol (HTTP). HTML is the language that allowed pages to display different fonts and sizes, pictures, colours etc.

It was Tim Berners Lee who brought this all together and created the World Wide Web. The first trials of the World Wide Web were at the CERN laboratories (one of Europe's largest research laboratories) in Switzerland in December 1990[26].

2.2 SEARCHING WWW:

The WWW is a software application that makes it easy and possible for anyone to publish and browse hypertext documents on the internet. With the advent of World Wide Web, it is no longer necessary to travel to the library to find the answer to a question as most of the information is available on the web but how this can be located? Earlier, unless users knew exactly where to look, they had trouble finding what they want. Unlike a library, the pages on the web were not as neatly organized as the books on the shelves and were not completely cataloged in one central location. Even knowing where to look for information (URL) is not
guarantee that we will find it. Usually, a forward address is provided for a page that has moved, but it may only be available for a short time [27].

Today there is a collection of search tools available that allows us to find information on the web quickly and easily. To organize and locate the information, two basic approaches have evolved. These are directories and search engines. In both approaches, information about web pages is contained in some database that has already been created either manually or using special programs called crawlers. Our request is answered by the search tool retrieving the information from its already constructed database of indexed web pages.

2.2.1 Directories:

The first method for finding and organizing Web information is the directory approach. A directory offers a hierarchical representation of hyperlinks to web pages and presentations broken down into topics and subtopics. Human editors usually review and classify the Web pages and presentations that are added to a directory. Directories can be classified as either general or specialized.

Web directories are directories where different resources are gathered. These are Similar to our desktop directories, where we gather files in a directory based on some criterion. Web directories are just large collections of links to sites, arranged in different categories. The sites in a Web directory are listed in some order (most often alphabetic) and users browse through them. Although many Web directories offer a search functionality of some kind, search directories are fundamentally different from search engines in the two ways – most directories are edited by humans and corresponding URLs are not gathered automatically by crawlers but submitted by site owners. The main advantage of Web directories is that there is a human to view and check the pages. Therefore, there is a lesser chance that pages will be classified in the wrong categories. The disadvantages are that the lists in web directories are sometimes outdated if no human was available to do the editing and checking for some time. Moreover, there is no crawler that means that we have to submit our URL to the search directory, rather than sit and wait for the spider to come to our site. The most popular general-purpose search directories are Google Directory and Yahoo! Directory.
Specialized directory is usually organized by an expert in a particular field. It offers a narrow selection of topics that have more depth. These are also known as subject guides. These directories deal with a variety of topics. If we want to know general information then general web directory will provide us the answer and if we want to know some specific information then specialized directory will locate the information more efficiently and quickly. For example shareware.com indexes over 190,000 software files located in archives around the Internet, including freeware, shareware, demos, patches, fixes, and upgrades. Users can search by platform, archive, or software name. Browsing by platform, archive, or keyword is also available. Approximate download times are given and download reliability is also noted. The site offers an e-mail newsletter to keep current on new additions. Completeplanet.com is a subject guide that provides access to invisible web (information that is not available to search engines). TravelGuru.com, Travelocity .com, yatra.com are some of the sites which are related to specific topic (travel). These sites provide us real time flight information and also the hotel booking for the places we want to visit.

2.2.2 Search Engine:

The second approach to organize and locate information on the web is a search engine like Google, Alta vista etc.(see figure 2.1). It is a computer program that does the following:

1. Allows user to submit a query consisting of a word or phrase describing the specific information he/she wants to search on the web.
2. Searches the database to match the query.
3. Collects and returns a list of clickable URLs that match the query.
4. Permits user to revise and resubmit a query.

A Search Engine is a program that is designed to search information on World Wide Web. It searches documents for specified keywords and returns a list of the documents often called hits where the keywords were found. The information may consist of web pages, images and other types of files. Some search engines also mine data available in databases or open directories. Without sophisticated search engines, it would be virtually impossible to locate
anything on the Web without knowing a specific URL. The next section discusses search engine in detail.

2.3 WEB SEARCH ENGINE:

There are basically three types of search engines: Crawler based search engines, Human powered search engines and Hybrid search engines [28].

Crawler-based search engines (like Google) are those in which crawlers (software programs) visit a Web site, read the information on that site, read the site's meta tags and downloads the documents. It also follows the hyperlinks that the document connects to. The crawler returns all that information back to a central repository of the search engine, where the data is indexed.

Human-powered search engines, better known as Web directories, are popular because of the higher quality of links submitted by humans and these links are indexed and catalogued [28]. The information that is submitted is only put into the index. Some of the most popular human-powered search engines on the Web are Google directory, Yahoo directory, Open directory. The Yahoo Directory is one of the oldest directories on the Web. It is a human-created and maintained library of web sites organized into categories and subcategories. Yahoo editors review the sites to be included in the Directory, and to evaluate the best place to list them.

Hybrid Search engine is a cross between crawler-based and human-powered directories. When we search using the hybrid method, both types (crawler and human powered) are
featured in the results. Usually, a hybrid search engine will favor one type of listings over another. For example, MSN Search is more likely to present human-powered listings from LookSmart. However, it also presents crawler-based results, especially for more obscure queries.

2.3.1 How Search Engine works:

A search engine is a program designed to find the information stored on a computer system such as the World Wide Web. The search engine allows one to ask for content meeting specific criteria and retrieving a list of references that match those criteria [28]. A search engine has three components:

1. Web crawler that finds and fetches the web pages.
2. Indexer that sorts every word on every page and stores the resulting index of words in a huge database.
3. The query processor, which compares the search query to the index and returns the most relevant documents.

The architecture of search engine is shown below in figure 2.2. It contains a front-end process and a back-end process. In the front-end process, the user enters the search words into the search engine interface, which is usually a Web page with an input box.

![Search Engine Architecture](image-url)
The application then parses the search request into a form that the search engine can understand, and then the search engine executes the search operation on the index files. After ranking, the search engine interface returns the search results to the user. In the back-end process, a crawler fetches the Web pages from the Internet, and then the indexing subsystem parses the Web pages and stores them into the index files.

1. **Web Crawler:**

A *Web crawler* is a computer program that traverses the World Wide Web in an automated manner or with a view to create a copy of all the visited pages for later processing by a search engine that will index the downloaded pages to provide fast searches [29]. Crawlers can also be used for automating maintenance tasks on a Web site, such as checking links or validating HTML code. It starts with a list of URLs to visit, called the *seeds*. As the crawler visits these URLs, it identifies all the hyperlinks in the page and adds them to the list of URLs to visit, called the *crawl frontier*. URLs from the frontier are recursively visited according to a set of policies. Architecture of a web crawler is shown in figure 2.3.

![Fig 2.3: Web Crawler Architecture](image)

The description of crawler modules shown in figure 2.3 is given below:

1. The URL frontier, containing URLs yet to be fetched in the current crawl.
2. A DNS resolution module that determines the IP address of web server from which to fetch the page specified by a URL.
3. A fetch module that uses the http protocol to retrieve the web page at a URL.
4. A parsing module that extracts the text and set of links from a fetched web page.
5. A duplicate elimination module that determines whether an extracted link is already in the URL frontier or has recently been fetched.

Crawling is performed by anywhere from one to potentially hundreds of threads, each of which loops through the logical cycle. These threads may be run in a single process, or be partitioned amongst multiple processes running at different nodes of a distributed system.

A crawler thread begins by taking a URL from the frontier and fetching the web page corresponding to the URL, generally using the http protocol. The fetched page is then written into a temporary store. Thereafter, the page is parsed and the text as well as the links in it are extracted. The text is passed on to the indexer. Link information including anchor text is also passed on to the indexer for the computation of rank of the document. In addition, each extracted link goes through a series of tests to determine whether the link should be added to the URL frontier.

Next, a URL filter is used to determine whether the extracted URL should be excluded from the frontier based on one of several tests. For instance, the crawl may seek to exclude certain domains. Many hosts on the Web place certain portions of their websites off-limits to crawling, under a standard known as the Robots Exclusion Protocol (Robot.txt) [30].

The robots.txt file must be fetched from a website in order to test whether the URL under consideration passes the robot restrictions, and can therefore be added to the URL frontier. Rather than fetch it afresh for testing on each URL to be added to the frontier, a cache can be used to obtain a recently fetched copy of the file for the host [30]. This is especially important since many of the links extracted from a page fall within the host from which the page was fetched and therefore can be tested against the host's robots.txt file. A URL (particularly one referring to a low-quality or rarely changing document) may be in the frontier for days or even weeks.

Next, a URL should be normalized. Often the HTML encoding of a link from a web page \( P \) indicates the target of that link relative to the page \( P \). Thus, there is a relative link encoded in the HTML of the page. For example:
Finally, the URL is checked for duplicate elimination: if the URL is already in the frontier or already crawled, it is not added to the frontier. When the URL is added to the frontier, it is assigned a priority based on which it is eventually removed from the frontier for fetching.

2. **Indexer:**

Data about web pages are stored in an index database for later search. The purpose of an index is to allow information to be found as quickly as possible [13]. A Search engine gives the indexer the full text of the pages it finds. These pages are stored in search engine’s index database. This index is sorted alphabetically by search term, with each index entry storing a list of documents in which the term appears and the location within the text where it occurs. This data structure allows rapid access to documents that contain user query terms. Search engine ignores common words called *stop words* (such as *the*, *is*, *on*, *or*, *of*, *how*, *why*). Stop words are so common that they do little to narrow a search, and therefore they can safely be discarded. The indexer also ignores some punctuation and multiple spaces.

The indexing process takes the detailed information collected by the crawler and analyses it. The indexed information is saved in a database, waiting for someone to do a search. When someone does a search at the search engine (or directory), the keywords entered in the search box are compared with the indexed information in the search engine's database and the list of corresponding web pages it feels are most relevant to the search are returned. There are two important methods of indexing used in web database creation - full-text indexing and human indexing as discussed below.

**Full-Text Indexing:**

As its name implies, full-text indexing is where every word on the page is put into a database for searching. Alta Vista, Google, Infoseek, Excite are examples of full-text databases. After parsing, the indexer adds the referenced document to the document list for the appropriate
words. The process of indexing is commonly split up into two parts, the development of a forward index and a process which sorts the contents of the forward index into the inverted index. The inverted index is so named because it is an inversion of the forward index. The forward index stores a list of words for each document as shown in table 2.1.

<table>
<thead>
<tr>
<th>Document</th>
<th>Words</th>
</tr>
</thead>
<tbody>
<tr>
<td>Document 1</td>
<td>the, cow, says, moo</td>
</tr>
<tr>
<td>Document 2</td>
<td>the, cat, and, the, hat</td>
</tr>
<tr>
<td>Document 3</td>
<td>the, dish, ran, away, with, the, fork</td>
</tr>
</tbody>
</table>

The principle behind developing a forward index is that as documents are parsed, the words corresponding to the document are stored as shown in table 2.1. It is essentially a list of pairs consisting of a document and a word, ordered by the document. Thereafter, the forward index is transformed to an inverted index. Converting the forward index to an inverted index is only a matter of sorting the pairs by the words. In fact, the inverted index is a word-sorted forward index.

Many search engines maintain an inverted index. When evaluating a search query, it quickly locates documents containing the words in a query. Generally, these documents are ranked by relevance. Because the inverted index stores a list of the documents containing each word, the search engine can use direct access to find the documents associated with each word in the query in order to retrieve the matching documents quickly. The table 2.2 shows a simplified illustration of an inverted index.

<table>
<thead>
<tr>
<th>Word</th>
<th>Documents</th>
</tr>
</thead>
<tbody>
<tr>
<td>the</td>
<td>Document 1, Document 3, Document 4, Document 5</td>
</tr>
<tr>
<td>cow</td>
<td>Document 2, Document 3, Document 4</td>
</tr>
<tr>
<td>says</td>
<td>Document 5</td>
</tr>
</tbody>
</table>
Human Indexing:

In the Full text indexing, all of the work was done by a computer program called a "spider" or a “crawler”. On the contrary, in human indexing, a person examines the page and determines a very few key phrases that describe it. This allows the user to find a good start of works on a topic assuming that the topic was picked by the human as something that describes the page. This is how the directory-based web databases are developed. Yahoo and some of Magellan are two of the few examples of human indexing.

3. Query Processor:

Query processor compares the search query to the index and returns the most relevant documents (see figure 2.4). It has seven possible steps [32]. Document processing shares many steps with query processing. More steps and more documents make the process more expensive for processing in terms of computational resources and responsiveness. However, longer the wait for results, higher is the quality of results. Hence, search system designers choose what is most important to their users — time or quality. Publicly available search engines usually choose time over the high quality as there is a large number of documents to search against. The steps in query processing are as follows:

- Tokenize query terms.
- Recognize query terms vs. special operators.
- Delete stop words.
- Stem words.
- Create query representation.
- Expand query terms.
- Compute weights.

**Tokenizing** - As soon as a user inputs a query, the search engine — whether a keyword-based system or a full natural language processing (NLP) system — tokenizes the query stream, i.e., breaks it down into understandable segments. Usually a token is defined as an alpha-numeric string that occurs between white space and/or punctuation.

**Parsing** - Since users may employ special operators in their query, including Boolean, adjacency, or proximity operators, the system needs to parse the query first into query terms and operators. These operators may occur in the form of reserved punctuation (e.g., quotation marks) or reserved terms in specialized format (e.g., AND, OR). In the case of an NLP system, the query processor will recognize the operators implicitly in the language used no matter how the operators might be expressed (e.g., prepositions, conjunctions, ordering). At this point, a search engine may take the list of query terms and search them against the inverted file. In fact, this is the level at which the majority of publicly available search engines perform the search.

**Stop list and stemming** - Some search engines not only stem the query, but also create a stop-list. The stop list might also contain words from commonly occurring querying phrases, such as, "I'd like information about." However, since most publicly available search engines encourage very short queries, as evidenced by the size of query window provided, the engines may drop these two steps.

**Creating the query** - How each particular search engine creates a query representation depends on how well it’s matching process works. For instance, if a statistically based matcher is used, then the query must match the statistical representations of the documents in the system. Good statistical queries should contain many synonyms and other terms in order to create a full representation. If a Boolean matcher is utilized, then the system must create logical sets of the terms connected by AND, OR, or NOT.

An NLP system will recognize single terms, phrases, and Named Entities. If it uses any Boolean logic, it will also recognize the logical operators from Step 2 and create a
representation containing logical sets of the terms to be AND'd, OR'd, or NOT'd. At this point, a search engine may take the query representation and perform the search against the inverted file.

**Query expansion** - Since users of search engines usually include only a single statement of their information needs in a query, it becomes highly probable that the information they need may be expressed using synonyms, rather than the exact query terms, in the documents which the search engine searches against. Therefore, more sophisticated systems may expand the query into all possible synonymous terms and perhaps even broader and narrower terms. This process approaches what search intermediaries did for end users in the earlier days of commercial search systems. Back then, intermediaries might have used the same controlled vocabulary or thesaurus used by the indexers who assigned subject descriptors to documents. Today, resources such as WordNet are generally available, or specialized expansion facilities may take the initial query and enlarge it by adding associated vocabulary.

**Query term weighing (assuming more than one query term)** - The final step in query processing involves computing weights for the terms in the query. Sometimes the user controls this step by indicating either how much to weight each term or simply which term or concept in the query matters most and must appear in each retrieved document to ensure relevance. Leaving the weighting up to the user is not common, because research has shown that users are not particularly good at determining the relative importance of terms in their queries. They can't make this determination for several reasons. First, they don't know what else exists in the database, and document terms are weighted by being compared to the database as a whole. Second, most users seek information about an unfamiliar subject, so they may not know the correct terminology. Few search engines implement system-based query weighting, but some do an implicit weighting by treating the first term(s) in a query as having higher significance. The engines use this information to provide a list of documents/pages to the user. Finally, the expanded, weighted query is searched against the inverted file of documents.
2.4 PERFORMANCE MEASURES:

*Information retrieval (IR)* searches information and metadata within documents, relational databases and WWW. There is an overlap in the usage of the terms: data retrieval, document retrieval, information retrieval, and text retrieval, but each term also has its own body of literature, theory and technologies. However, an information retrieval process begins when a user enters a query into the system where queries are formal statements of information needs submitted in the form of simple search strings in web search engines. User queries are matched against the database information. The query does not uniquely identify a single object in the collection. Instead, several objects may match the query, perhaps with different degrees of relevancy where an object is an entity that is represented by information in a database. Depending on the application the data objects may be, for example, text documents, images, audio or videos. Most IR systems compute a numeric score on how well each object in the database match the query, and rank the objects according to this value. The top ranking objects are then shown to the user. The process may then be iterated if the user wishes to refine the query.

Many different “measures” are employed for the evaluation of the performance of information retrieval systems [35, 36, 37, 38]. Given a collection of documents and a query, these measures help in identifying a document as relevant or non-relevant to a particular query. Three common measures are precision, recall and accuracy.

2.4.1 Precision:

Precision is the fraction of the documents retrieved that are relevant to the user's information need.

\[
\text{Precision} = \frac{(\text{relevant documents}) \cap (\text{retrieved documents})}{(\text{retrieved documents})} \quad (2.1)
\]
2.4.2 Recall:

Recall is the fraction of the documents that are relevant to the query that are successfully retrieved.

\[
\text{Recall} = \frac{(\text{relevant documents}) \cap (\text{retrieved documents})}{(\text{relevant documents})} \quad (2.2)
\]

For example, in case of text search on a set of documents, recall is the number of correct results divided by the number of results that should have been returned. It can be looked at as the probability that a relevant document is retrieved by the query. In the context of classification tasks, the terms true positives, true negatives, false positives and false negatives are used. Where

1. tn / True Negative: case was negative and predicted negative
2. tp / True Positive: case was positive and predicted positive
3. fn / False Negative: case was positive but predicted negative
4. fp / False Positive: case was negative but predicted positive

High precision means that everything returned was a relevant result.

\[
\text{Precision} = \frac{tp}{tp + fp} \quad (2.3)
\]

In even simpler terms, a high recall means you haven't missed anything.

\[
\text{Recall} = \frac{tp}{tp + fn} \quad (2.4)
\]

2.4.3 Accuracy:

It is defined as the degree of closeness of measurements of a quantity to that quantity's actual (true) value.
\[
\text{Accuracy} = \frac{tp + tn}{tp + tn + fp + fn}
\]  

(2.5)

The index maintained at search engine site is like a library card catalog where the information is organized carefully by librarians for easy access by users. To find the information on the web, we enter the appropriate query in terms of words into search box of search engine and it will return the information that we want. However, if we are looking for something specific like mathematics book in the library, we cannot expect the book lying on the librarian’s front desk. We have to excavate it from library book racks. This is the situation on the web where traditional search engines will not help us. Moreover, Search engines only search a small portion of web making the Hidden Web a potential resource to explore. In fact, World Wide Web (WWW) can be broadly divided into following categories [37]:

- **Surface Web** contains static web pages and these pages are linked to many other pages. Traditional search engines create their indices by crawling these web pages. However, The Surface web contains 1% of information content of the web. Search engine crawl along the hyperlinks to extract and index text from HTML documents lying on the websites. Thereafter, the information is made searchable through keywords.

- **Hidden Web** consists of web pages that are created dynamically by filling the search query forms. The Hidden web contains 99% of information content of the web. Most of this information is contained in the databases and hidden behind search interface and therefore is not indexed by traditional search engines.

This means if we are searching for information from surface web only, we are searching through only 1% of the total resources and missing 99% of it whereas 95% of hidden web consists of freely accessible information.
2.5 WHAT IS HIDDEN WEB AND WHY IS IT HIDDEN:

The Hidden Web refers to the largest sector of online information resources on the Internet, and yet the first barrier in discussing it is its name. Why this part of web is known as hidden? The answer also lies inside its name. Hidden web is the term used to describe the information available on the web that is hidden behind the search query interfaces that act as entrance to backend databases. Unlike pages on the Surface Web (that is, the Web that can be accessed from search engines and directories), information in databases is generally inaccessible to the crawlers that create search engine indexes.

Hidden web data, stored in structured or unstructured databases [1], is inherently hidden behind search forms. It is qualitatively and quantitatively different from the Surface Web. The quality content of the Hidden Web is 1,000 to 2,000 times greater than that of the Surface Web whereas overall the hidden web contains approximately 7,500 terabytes of data and 550 billion individual documents in contrast to the Surface Web, which is reported to about 167 terabytes [1]. Since the hidden web is the biggest source for structured data and is not publically indexed yet, accessing the same is a challenging task especially when the pages are created dynamically through search interfaces. On average, Hidden websites receive fifty per cent greater monthly traffic than surface websites. In recent years, a rapid growth of databases has been witnessed on the Web, or the so called “Hidden Web.” A survey [1] estimated that 96,000 “search sites” and 550 billion content pages in this hidden web. [7] estimated that there are currently 450,000 online databases. With the virtually unlimited amount of information sources, the Hidden Web is clearly an important frontier for data integration.

Several online databases provide dynamic query-based data access through their query interfaces, instead of static URL links [40, 43]. This Query interface is considered as an entrance to Hidden Web. For example, if user wants to buy a second hand car, he has to fill the search box of search engine by his query. It will get back to him with many result indexes that contain search forms. Now, user has to fill the form according to his specifications and will get the desired result. However, a traditional crawler cannot fill the form on user’s behalf.
and moreover, there is no mechanism for the crawler to go inside the database tables and extract the data. Hence, database content is therefore "hidden" to user.

2.6 WHY HIDDEN WEB CANNOT BE INDEXED BY TRADITIONAL SEARCH ENGINES:

Surfacing the Hidden Web is more difficult task in many respects. First, the index structures for the hidden web deal with the structured as well as the large volume of data. Second, the Search query interfaces often have more than one attribute and require their respective values to be submitted. Since hidden web pages are created dynamically by firing user query, they cannot be indexed by traditional search engines [8]. There are many reasons behind this and some of them are:

- There are some pages that aren’t submitted directly to search engines via using “submit URL” feature and many pages do not have back links pointing to them from other pages. Thus, there is no way for conventional search engines to find them. According to CNNIC[48], 37.6% of pages do not link to any web page.

- Hidden Web data is stored in backend relational databases. This is the technical barrier to search engines for extracting information from such resources. The major problem being that a crawler cannot interact with them like human beings can. For instance, Hidden web databases may contain 100,000 unique records requiring 100,000 direct queries to be issued to that database to obtain all records contained therein. The problem aggravates when there are tens of thousands of such hidden web databases available on the web.

- Private Web which is a kind of Hidden Web, consists of personal unpublished information and most of webmasters do not want search engines to index their pages. In fact, the pages are protected by thewebmaster:by writing appropriate entries into "robots.text" file and "noindex" or "nocache" meta tags in the HTML code of a web page. Web sites related to commercial profits, state secret and military deployments are some examples of this kind.
• Many websites restrict their access to some pages of the websites or require a password and is open to registered users who have passed the verification. Some websites charge no registration fee, but some ask for payment. Even the free registration forms become obstacles that crawlers cannot fill by their own and this web naturally can’t be indexed by conventional search engines.
• Pages with only JavaScript or Flash-based content cannot be easily indexed.
• Some websites like news, stock market, flight etc. have real time constraints. The information in these sites bear strong timeliness, and there is no use once the information becomes stale. So, these sites should be updated frequently (about once in 20 days, monthly or even daily).

2.7 ENTRANCE TO HIDDEN WEB DATABASES:

Search query interfaces are considered as entrance to Hidden web databases. It consists of html forms. They are designed for human understanding to query the database. It contains a sequence of interface components, i.e., text-labels and form elements (textbox, radio buttons, selection list, check box etc.). It does not have a standard layout of components and there is infinite number of possible layout patterns [49]. A label is usually associated with an element to describe the semantic meaning of the element. Logically, elements and their associated labels together form different attributes of the underlying database. Using such a schema-based interface, users can submit appropriate queries to Web databases. It also contains multiple attributes. For example, there are 3 attributes in the interface of the car domain as shown in figure 2.5 and the respective source code is shown in figure2.6.

![Interface of www.autonagar.com](image)
Each attribute consists of three components: label, (internal) name, and domain. Where

- **Label**: The label of an attribute is a piece of text on the query interface, which denotes the meaning of the attribute to the user. For example, the first attribute in figure 2.5 has a label “Select Make”.

- **Name**: The name of an attribute is the internal name of the attribute given in the HTML script for the identification purpose. For example, the name of attribute Select Make is “make”.

- **Domain**: The domain of an attribute is a set of values the attribute may take. For example, the domain of the attribute Select Make is \{Audi, BMW, ........., Hindustan Motors\}.

First step of Hidden Web extraction process is the detection of Hidden Web search interface. It becomes one of the most important parts of hidden web information retrieval. Some of the important hidden web search interface detection techniques are discovered below.
One of the hidden web crawler architecture was proposed by Sriram Raghavan and Hector Garcia- Molina [50]. In this paper, a task-specific, human-assisted approach is used to crawl the hidden web. In this research work, a model of task specific human assisted web crawler called HiWE was designed and realized. The HiWE prototype was built at Stanford. It is designed to automatically process, analyze, and submit forms, using an internal model of forms and form submissions. In fact, it uses a layout-based information extraction (LITE) technique to process and extract useful information. The advantages of this architecture is that it is a task specific approach and allows the crawler to concentrate on relevant pages only and automatic form filling can be done with human assistance. Limitations of this architecture are that it is not precise with regards to partially filled forms and is not able to identify and respond to simple dependencies between form elements.

A technique for collecting hidden web pages for data extraction is proposed by Juliano Palmieri Lage et al. [51]. In this technique the authors have proposed the concept of web wrappers. A web wrapper is a program which extracts the unstructured data from web pages. It takes a set of target pages from the web source as an input. These set of target pages are generated by an approach called “Spiders” which automatically traverse the web for web pages. Hidden web agents assist the wrappers to deal with the data available on the hidden web. The advantage of this technique is that it can access a large number of web sites of diverse domains. However, the limitation of this technique is that it can access only that web site that follows common navigation patterns. Further, modification can be done in this technique to cover navigation patterns based on these mechanisms.

A technique for automated discovery of search interface from a set of html forms is proposed by Jared Cope, Nick Craswell and David Hawking [52]. This paper defined a novel technique to automatically detect search interface from a group of html forms. A decision tree was developed with the C4.5 learning algorithm using automatically generated features from html markup that can give a classification accuracy of about 85% for general web interfaces. Advantage of this technique is that it can automatically discover the search interface. Limitation of this technique is that it is based on single tree classification method and number of feature generation is limited due to use of limited data set.
A technique for understanding web query interfaces through best effort parsing with hidden syntax is proposed by Zhen Zhang et al. [53]. It addresses the problem of understanding web search interfaces by presenting a best-effort parsing framework. This work presented a form extractor framework based on 2P grammar and the best effort parses in a language parsing framework. It identifies the search interface by continuously producing fresh instances by applying productions until attaining a fix-point, when no fresh instance can be produced. Best effort parser technique minimizes wrong interpretation as much as possible in a very fast manner. It also understands the interface to a large extent. Advantage of this technique is that it is a very simple and consistent technique with no priority among preferences and it can handle missing elements in a form. The limitation of this technique is that establishment of single global grammar that can be interacted to the machine globally.

A technique named as “siphoning hidden web data through key word based interface” for retrieval of information from hidden web databases through generation of a small set of representative keywords and build queries is proposed by Luciano Barbosa and Juliana Freire [54]. This technique is designed to enhance coverage of deep web. Advantage of this technique is that it is a simple and completely automated strategy that can be quite effective in practice, leading to very high coverage of hidden web. Limitation of this technique is that it is not able to achieve the coverage for collection whose search interface fixes a number of results. Further the authors have advised that modification can be done in this algorithm to characterize search interfaces techniques in a better way so that different notions and levels of security can be achieved.

An improved version of random forest algorithm is proposed by Deng et al. [55]. In this improved technique a weighted feature selection algorithm is proposed to generate the decision trees. The advantage of this improved algorithm is that it minimizes the problem of classification of high dimension and sparse search interface using the ensemble of decision trees. Disadvantage of this improved algorithm is that it is highly sensitive towards the changes in training data set.

Further improvement in random forest algorithm is done by Yunming Ye et al. [56] by using feature weighting random forest algorithm for detection of hidden web search interface. This
work presents a feature weighting selection process rather than random selection process. Advantage of this technique is that it makes a weighted feature selection process instead of random selection. Hence, it reduces the chances of noisy feature selection. However, the limitation of this technique is that features are only available in the search forms.

An algorithm named as “The naive bayesian web text classification algorithm” is proposed by Ping Bai and Junqing Li [57] for automatic and effective classification of web pages with reference to given model for machine learning. In the conventional techniques, category abstracts are produced using the inspection by domain experts either through semiautomatic method or artificial method. All the items provided are equally important according to conventional common bayesian classifier whereas according to improved naive bayesian web text classification algorithm, whole of the items in every title are provided higher importance to others. The strength of this technique is that text classification results are very accurate and further scope in this algorithm is suggested to make the classification process automatic in an efficient way.

An approach for automatic detection and unification of web search query interfaces using domain ontology is proposed by Anuradha and A.K.Sharma [58]. The technique works by concentrating the crawler on the given domain considering the domain ontology. This technique automatically detects the domain specific search interfaces by searching the domain features in the source code of the page. Domain ontology table has been built with domain word, its synonyms, attributes of the interface and the respective synonyms, to separate the search interfaces of particular domain form other pages. The strengths of this technique are that results are produced from multiple sources, human effort is reduced and results are very accurate in less execution time. Limitation of this technique is that it is domain specific.

2.8 VARIOUS TECHNIQUES FOR DETECTION OF HIDDEN WEB SEARCH INTERFACES:

By going through some of the hidden web search interface detection techniques discussed above, it is concluded that each technique for detection of Hidden web search interface has
some relative strengths and limitations. A tabular summary is given below in the table 2.3, which summarizes the techniques, strengths and limitations of some of important detection techniques for hidden web search interface.

Table 2.3: Summary of various techniques for detection of Hidden web search interfaces

<table>
<thead>
<tr>
<th>Authors</th>
<th>Technique</th>
<th>Strengths</th>
<th>Limitations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sri Ram Raghavan et al. (2001)</td>
<td>Hidden Web Exposer</td>
<td>An application specific approach to hidden web crawling</td>
<td>Imprecise in filling the forms.</td>
</tr>
<tr>
<td>Palmieri Lage et al. (2002)</td>
<td>Hidden Web Agents</td>
<td>Wide coverage of distinct domains.</td>
<td>Restricted to web sites that follow common navigation patterns.</td>
</tr>
<tr>
<td>Jared Cope et al. (2003)</td>
<td>Single tree classifiers</td>
<td>Automatically discovery of search interface, performed well when rules are generated on the same domain.</td>
<td>Long rules, large size of feature space in training samples, Over fitting, Classification precision is not very satisfying.</td>
</tr>
<tr>
<td>Zhen Zhang et al. (2004)</td>
<td>2P Grammar and Best effort Parser</td>
<td>Very simple and consistent, No priority among preferences, Handling of missing elements in form.</td>
<td>Critical to establish single global grammar that can be interacted to the machine globally.</td>
</tr>
<tr>
<td>Luciano Barbosa et al. (2004)</td>
<td>Automatic query generation based on small set of keywords.</td>
<td>A simple and completely automated strategy that can be quite effective in practice</td>
<td>A large domain of Keywords has to be generated.</td>
</tr>
<tr>
<td>Deng, X. B. et al. (2008)</td>
<td>weighted feature selection algorithm</td>
<td>Minimizes the problem of classification of high dimension and sparse search interface using the ensemble of decision trees</td>
<td>Highly sensitive towards the changes in training data set.</td>
</tr>
<tr>
<td>Ye, Li, Deng et al. (2009)</td>
<td>Feature weighted selection process</td>
<td>Minimizes the chances of selection of noisy features.</td>
<td>No use of contextual information associated with forms.</td>
</tr>
<tr>
<td>--------------------------</td>
<td>----------------------------------</td>
<td>--------------------------------------------------</td>
<td>----------------------------------------------------------</td>
</tr>
<tr>
<td>Ping Bai et al. (2009)</td>
<td>Naïve Bayesian Algorithm</td>
<td>Text classification results are very accurate.</td>
<td>Classification algorithm is not automatic.</td>
</tr>
<tr>
<td>Sharma et al. (2010)</td>
<td>Based on domain ontology</td>
<td>Results are produced from multiple sources, reduces the human effort, less execution time, accuracy is high.</td>
<td>It is domain specific.</td>
</tr>
</tbody>
</table>

Since Hidden web search interfaces are the entry points for the searching of the hidden web information, a hidden web crawler should understand and detect the search interfaces efficiently to retrieve the hidden web information. This detection technique should be designed in such a way that it should be automatic, compatible with current web technology and should have strengths in terms of wide coverage of different domains, resistant to unwanted features and should be user friendly approach.

### 2.9 HOW TO EXPLORE HIDDEN WEB:

In addition to what we find in search engine results, there are many other ways to explore the Hidden Web [69]. These are:

**2.9.1 Users Themselves:**

Users themselves should be aware of Hidden Web sites. He/she should increase his/her ability to extract, evaluate and use information among all kinds of resources. He should become active hunter but not the passive user. He/she should accumulate experiences in continuous practice and make full use of conventional search engines.
2.9.2 Make Use of Search Directories:

A directory offers a hierarchical representation of hyperlinks to web pages and presentations broken down into topics and subtopics. Some directories offer a gateway to hidden web. By using these gateways, we can find relevant databases and then use the database specific search tool to extract the information we want. Some of them are www.completeplanet.com, Invisible web directory, www.directsearch.com etc.

2.9.3 Modify Traditional Search Engine:

Many search engines only index HTML format files, and the majority of non-HTML files are not considered. Some powerful search engines like Google indexes non-HTML files such as PDF, DOC, PPT. Therefore, in order to index structured data and more types of files, traditional search engines must be improved.

2.9.4 Meta Search Engine:

A meta search engine or all-in-one search engine performs a search by consulting more than one search engines to do the actual work. It does not maintain its own database of information. While, submitting searches to other search engines, it queries the databases of other search engines. Some of them are Dogpile, MetaCrawler, Turbo10, Profusion. Advantage of this type of search engine is that we can access a number of different search engines with a single query. The disadvantage is that we will often have a lot of matches that will not be of our interest. But again this will return the list of various search forms of structured databases.

2.9.5 Develop a Hidden Web search engine:

In order to extract information from Hidden web, it is necessary to develop a special search engine that retrieve the hidden web content with a view to give high quality information to the web user in integrated form. This search engine should automatically discover domain
specific hidden web databases from the Web and then crawl and integrate the hidden web content by querying search interface forms. Upon user querying, search engine should search query from this integrated database by forwarding the search to right direction of domain. Several recent research projects, e.g., MetaQuerier [46] and WISE-Integrator [47], are exploring this exciting direction.

2.10 ONTOLOGY:

Domain ontology can be defined as information architecture in a specific domain. It provides a structured way of describing knowledge. It also defines the concepts about Web page categories and their hierarchical relationships. Ontology is a specification of conceptualization [73] that is description of concepts and their relationships. Ontology is composed of concepts, attributes and the relation among concepts. Concept is anything that can be described. It can be a real, assumed, concrete or abstract. A concept in ontology can be described by the attribute values. The main function of ontology is to provide the knowledge base needed for the classification of search results. Organizing search results into hierarchical structure can help users navigate, seek and find more quickly information they are looking for. Figure 2.7 provides an example of a university ontology describing the concept hierarchy in university domain. Pendidikan (education), Penelitian (research), Pengabdian (public service), Kemahasiswaan (student), and Fasilitas (facility) are sub-concepts of the university. Furthermore, leaf node concepts also have attribute values representing the characteristics of the corresponding concepts. In this example, komunitas (community), and masyarakat (people) are terms representing the concept of Pengabdian (public service). The basic building blocks of ontologies are concepts and relationships.

Concepts in the ontologies usually have a textual description defining them, although some ontologies include a formal definition in some kind of logic as well. In almost every ontology, concepts are described by one or more terms. Note that each concept might have more than one term describing it and that a term need not match only one concept. For example, to describe the concept of bicycle the terms “bicycle” and “bike” can be used. However, the term “bike” might also refer to the concept of motorcycle. Usually, ontologies include a single and unambiguous term for each concept.
This might be more appropriate for specifying and sharing knowledge, however, it is not usually good for detecting concepts in text because in real text the same concepts are usually referred to with many different terms. Relationships are usually of a specific type and connect two or more concepts.

2.11 SUMMARY:

It is almost impossible to predict what the Internet will be like in ten years time. There is, however, one quite interesting dark side of the Internet that has existed for some time, yet which very few people know about. This is something known as the Dark Net, Deep Net or Hidden Web.

To put it very simply, the web is defined as a collection of hyperlinks that are indexed by search engines. In other words, When we use a search engine like Google or Yahoo!, the information we get back is sometimes referred to as the "Surface Web", "Visible Web", or publically indexable web. However, there's a lot more information out there that Google and Yahoo can't find. That's the Hidden Web. The figure 2.8 summarizes the difference between Surface Web and Hidden Web.
Fig 2.8: Surface Web Vs Hidden Web

The Hidden Web consists of web pages and data that are beyond the reach of search engines. Some of what makes up the hidden Web consists of abandoned, inactive web pages, but the majority of data that lies within have been crafted to deliberately avoid detection in order to remain anonymous. A detailed discussion on Hidden Web Crawling techniques is given in next chapter.