Comparative Analysis of approaches for detecting near-duplicate URLs for search engine

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

Master of Engineering

In

Computer Science

Submitted by:

Shashank Panwar

(Roll No. 801432025)

Under the Supervision of:

Mr. Vinay Arora

(Assistant Professor)

Computer Science Engineering and Department

Thapar University

Patiala – 147004

June 2016

Certificate

I hereby certify that the work which is being presented in the thesis report entitled, **"Comparative Analysis of approaches for detecting near-duplicate URLs for search engine"**, submitted by me in partial fulfillment of the requirements for the award of the degree of Master of Engineering in Computer Science and Engineering at Computer Science and Engineering department of Thapar University, Patiala is an authentic record of my own work carried out under the supervision of *Mr. Vinay Arora* and refers other researcher's work which are duly listed in reference section.

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

(Shashank Panwar)

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

(Mr. Vinay Arora)
Assistant Professor
CSED, Thapar University
Patiala

Counter singed By

(Dr. Maninder Singh)
Head of Department
CSED, Thapar University
Patiala

(Dr. S. S. Bhatia)
Dean (Academic Affairs)
Thapar University
Patiala

Acknowledgement

I express my sincere and deep gratitude to my guide Mr. Vinay Arora, Assistant Professor, Computer Science and Engineering and Department, Thapar University, Patiala for the invaluable guidance, support and encouragement. He provided me all the resources and guidance throughout the thesis work.

I am also thankful to Dr. Maninder Singh, Head of department, Computer Science and Engineering Department for his kind help and cooperation. I express my gratitude to all the staff members of Computer Science and Engineering Department for providing me all the facilities required for the completion of my thesis work.

I would also like to express my appreciation to my good friend for the timely motivation and providing interesting environment. It was great pleasure in working with them during the thesis work.

Last but not the least I would like to thank God and my parents for not letting me down at the time of crisis and showing me the silver lining of the dark clouds.

(Shashank Panwar)
(801432025)

Abstract

The content on the web is increasing rapidly due to which the use of search engine is becoming vital for information retrieval. Search engine uses web crawler to traverse the web content available on the internet. Web crawler is an internet robot which visits web sites and fetches the content in order to create entries for search engine's index. Due to huge amount of web pages on the web there is a problem in front of search engine regarding removal of the duplicate and near-duplicate URLs. The duplicate detection techniques are divided into two main categories *viz.* Conventional approaches and Modernistic approaches. Conventional approaches include fingerprinting approach, shingling approach, cluster based approach, URL based approach, and keyword based approach. Modernistic approach includes locality sensitive hashing. In Conventional approaches, only fingerprinting approach *i.e.* MD5 signature is practically implemented in web crawler and all other approaches are just a concept or can be used in near future.

The comparative analysis of conventional approach and modernistic approach can be done on the basis of two parameters. First is time taken by the crawler to crawl the websites and second is number of duplicate document detected by the crawler using different algorithms. The process of removing the duplicate copies of the particular content is called Deduplication.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 World Wide Web

The World Wide Web (WWW) is a vast client-server system with millions of servers distributed across the world. A collection of web pages is maintained by each server. A client send the request to the server for a particular page, the server fetches the page from local file and revert it to the client. Further, server can also accept requests for storing new websites. A Uniform Resource Locator (URL) can be used to locate a web page on the WWW. It gives the location where the document is stored, often by embed the Domain Name Server (DNS) name with its associated server along with a file name by which the server can look up the page in its local file system. The application-level protocol is specified by a URL for transferring the web page across the network.

Application software, known as web browser, is used by the client to interact with the web servers and responsible for properly displaying a web document. An input is accepted by a web browser from a user as an URL. The overall organization of the web is shown in fig 1.1.

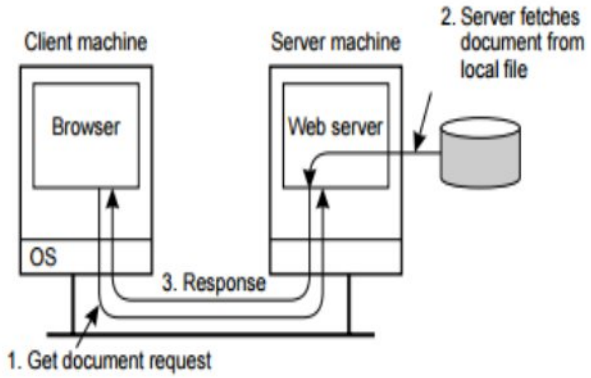

Fig 1.1: Organization of the web

The web pages on the WWW are written in a language called Hyper Text Markup Language (HTML), which supports hypertext means links to other web pages, video, audio, and graphics files. WWW is categorizing into two main categories, first is

structural component and second is semantic component. Structural component consists of browsers, servers, caches and internet. Semantic component consists of URLs, HTML and Hyper Text Transfer Protocol (HTTP).

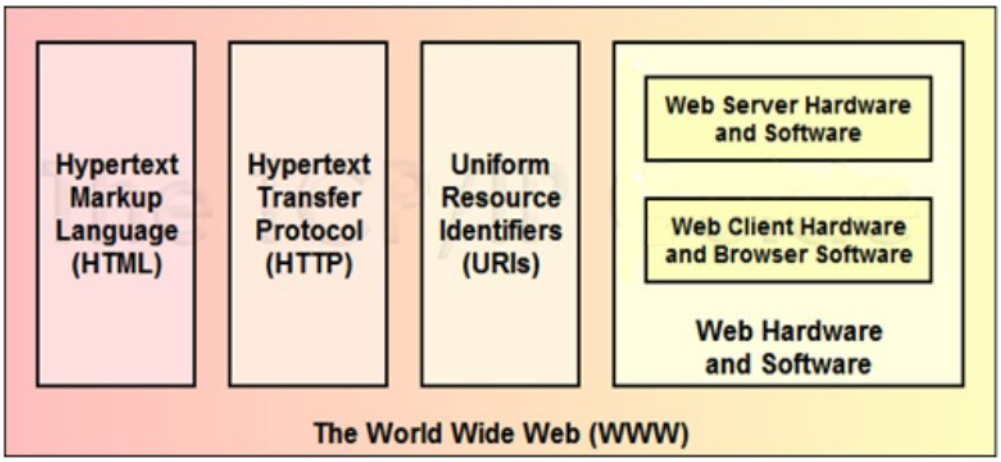

Fig 1.2: Functional Components of the World Wide Web

1.2 Working of Internet

The Internet is a worldwide network connecting millions of computers. The Internet is the global interconnected wireless, personal and mainframe, computer networks which uses the Internet protocol suite (TCP/IP) to link millions of devices worldwide. The Internet is a system having two main components: hardware components and protocol components. Hardware components consists of clients ,which is the end-point users and servers is a machine that stores the information we seek on the internet and nodes which serve as a connecting point along a route of traffic. Protocol components consist of set of protocol. The two most important protocols for accessing the Internet are transmission control protocol (TCP) and Internet protocol (IP).

Network Service Providers (NSPs) are the backbone of internet which contributes various networks which are connected to each other. Some well known NSPs are PSINet, SprintNet, IBM and others. Each NSP is connected to other NSP using Network Access Point (NAP) with the help of which packet traffic may jump from one NSP's backbone to another NSP's backbone. NSPs also connected with Metropolitan Area Exchanges (MAEs). The major difference between MAEs and NAPs is that NAPs are privately owned. NAPs are original Internet interconnects

points. Both MAEs and NAPs are known as Internet Exchange Points or IXs. Small network provider, *i.e.* ISPs and small bandwidth provider, purchase the bandwidth from NSPs.

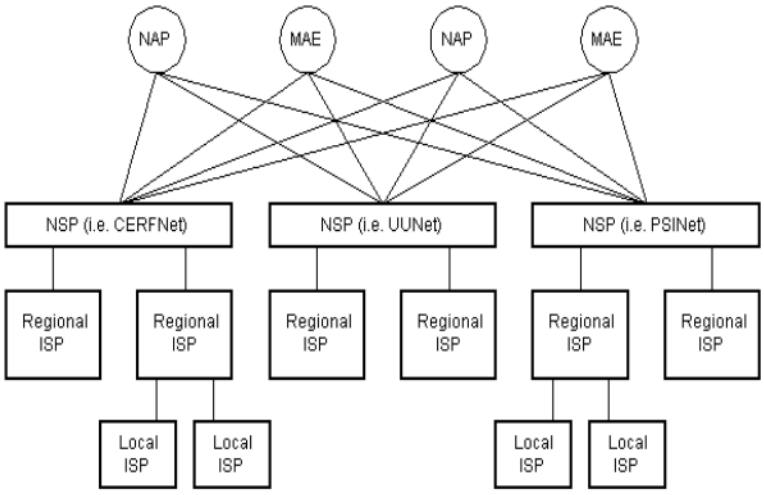

Fig 1.3: Hierarchical infrastructure of the Internet

1.3 Search Engine

A software system designed for searching information on the WWW known as search engine. The results given by the search engine often referred as Search Engine Result Pages (SERPs). The information might be a videos, images, web pages, *etc*. Some search engines are Yahoo, Bing, and Google *etc* which enable users to explore information on the web.

Fig 1.4: Cyclic architecture of search engine

Search engine follows cataract design in which operations are executed in strict order: web crawling, indexing and searching. The typical cataract model is depicted with thick arrow in the above fig 1.4.

1.3.1 Working of Search Engine

Search engine is a program which searches information in a database based on characters or keywords specified by the users. It mainly used for finding particular sites on the WWW.

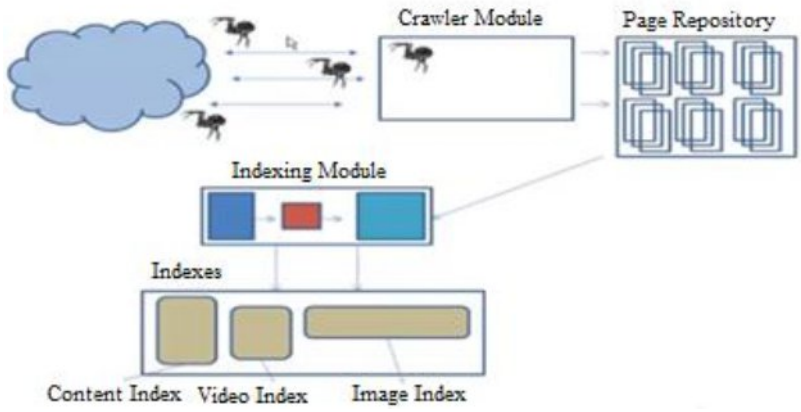

Fig 1.5: Component of Search Engine

There are varieties of search engines works in different ways, but among these all maintain the following process on real time:

i. **Web Crawling:** Web crawling is a process by which a search engine gathers the information from the web. A web crawler can be used for web crawling. The Crawler checks for the standard filename robot.txt, also known as robots exclusion standard or robot exclusion protocol, before sending certain information back to the indexed. The robots.txt is a standard used by websites to communicate with crawlers and other web bots. The standard informs the web bots about which areas of the website should not be processed or scanned. The URLs are extracted by the crawler from the retrieved pages and send this information to the URL frontier which determines what links to visit next and feed the links to visit back to the crawler.

ii. **Page Repository:** As shown in fig. 1.5 data extracted from the web by the crawler is stored in a database known as page repository. Search is performed through page repository and needs to be updated frequently. During or after completing crawling process search engine must store all the pages which are retrieved from

the web. Sometimes search engine maintains pages act as a cache for accessing frequently used web pages. These pages allow search engine to serve results pages very quickly, in addition to providing basic search facilities.

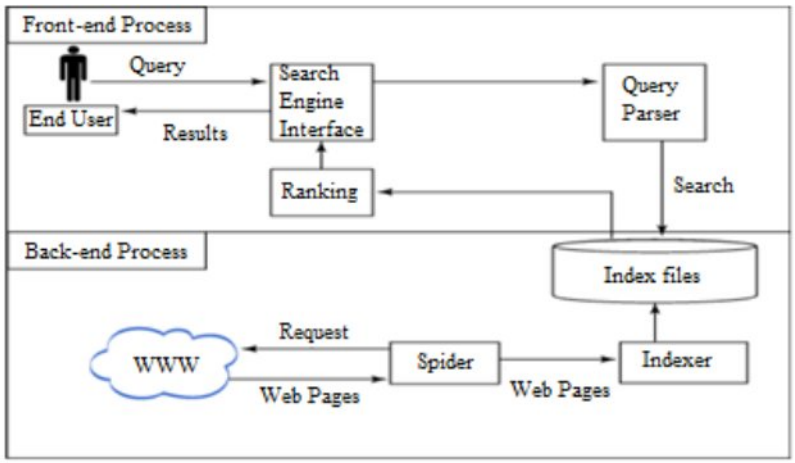

Fig 1.6: Search engine working steps

iii. **Indexing:** This is the process of creating a index of the web pages retrieved by the web crawler. After the web page stored in the page repository the job of search engine is to make an index for fast retrieval. All the words are extracted from the page repository by the indexer module and record the corresponding URL of words, a large “lookup table” is formed. Look up table can provide all the URLs which point to pages where a given word occurs. The indexing constitutes special difficulties, due to rapid rate of change of pages and vast size of internet.

iv. **Searching:** When a client or user enters a query into a search engine it may be a set of words or single word. Search engine find the specified words into the lookup table. The lookup table already contains the URLs corresponding to the specified word and the results obtained.

v. **Querying:** Handling of user’s search request is the responsibility of the query parser. The search engine heavily depends on the page repository and indexed.

vi. Ranking Algorithm: Search engine uses this algorithm to decide the importance of web pages in the search results. PageRank was developed by Sergey Brin and

Larry Page at Stanford University in 1996 as a part of research project [1]. The ranking algorithm has been used by the Google is PageRank. Google's definition, PageRank consider links to be like votes. PageRank is Google's system of counting link votes and determining which pages are most important based on them. The fundamental assumption is that if a website has more links from other website then the site is more important. PageRank is a link analysis algorithm and assigns a weight to each element of a hyperlink set of documents, such as the World Wide Web, with the purpose of measuring the relative importance within the set.

Beyond simple keyword lookups, search engine offers their own Graphical User Interface (GUI) or command driven operators and parameters for search to refine the search results. Google is the world's most popular search engine, having a market share of 69.89%. Bing comes in at second place with 11.94% market share and Yahoo! Comes in third place with 7.78% of market share as on June 2016 [2]

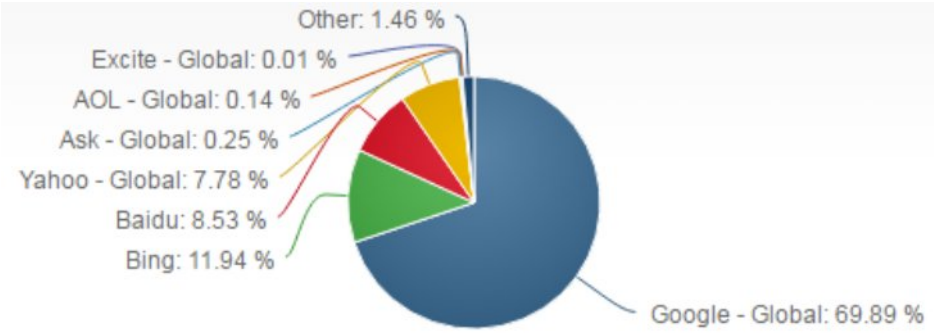

Fig. 1.7: Desktop search engine market share[2]

1.4 Web Crawler

As discussed earlier section, Web crawling is one of the most vital jobs of any search engine and is done by web crawler. The detailed structure of web crawler will be discussed in this section.

1.4.1 Definition of Web-Crawler

Web Crawler, also known as Crawler, Spider, Robot (or bots), Web agent, Wanderer or worm, is an automated script or a program which systematically

examines the Web. The WWW is browsed by Web crawler, typically for the purpose of Web indexing. Words in documents are indexed by the web crawler and also add those words to a database. Only those sites are visited by the crawler whose owner declares it as updated or new. Crawler visits and indexes selected web pages or entire web sites. Scooter is the crawler for AltaVista search engine . Crawler can be used to gather information from web pages, such as harvesting e-mail address. Automatic maintenance tasks on a website can also be performed by the crawler such as validate HTML code or checking links. If a crawler can assess whether a newly crawled web page is a near duplicate of a previously crawled web page or not then it will definitely increase the quality of a crawler.

1.4.2 Essential features of web crawler

i. **Robustness:** Web server creates spider traps, which generates web pages that misleads crawlers and stuck into fetching infinite number of pages in that domain. The design of crawler must be flexible to handle such traps. Every trap is not malicious; some are the unintentional side-effect of faulty website development.

ii. **Politeness:** Web crawler has both explicit and implicit policies which regulates the visiting rate to the websites. If a crawler visits a website again and again in a short period of time then the load of the server is increase. The solution for this problem is robots exclusion protocol [3].

1.4.3 Web Crawler architecture

Web crawler consists of four modules. These modules are shown in fig 1.8.

- Processing Queue: This queue is user for storing the URLs after parsing web document.
- Scheduler is used to schedule the URLs present in processing queue for downloading and parsing.
- Downloader is used for download the content of the given URL and the parse the document to extract set of URLs and update processing queue.

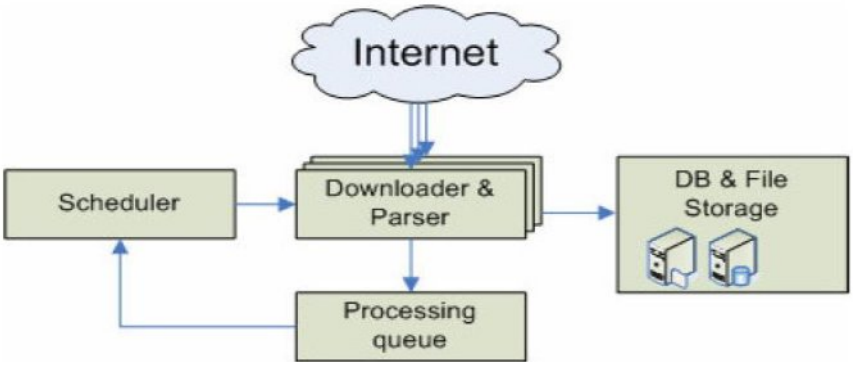

Fig 1.8: Web Crawler components

- Database and File Storage are used to store web document for further processing.

1.4.4 Working of Web Crawler

The working of web crawler consists of several modules which fit together to form a whole crawler system. Several modules are as follows:

i. **URL frontier**: A list of URLs is started to visit by the crawler, that list is known as seeds. When the crawler visits these URLs, it identifies all the hyperlinks in the page and adds them to the list of URLs for further crawling, called the crawl frontier. Crawler recursively visits the URLs from the frontier according to a set of policies. Two important factors manage the order in which URLs are returned by the frontier. First, politeness means do not hit a web server too frequently. Second, freshness means crawl some pages more often than others e.g. pages (such as news sites) whose content change often.

ii. **DNS resolution**: DNS resolution is the process of translating web page URL such as www.wikipedia.com into an IP address. Whether you're accessing a Web site or sending e-mail, your computer uses a DNS server to look up the domain name you're trying to access.

iii. **Parsing module:** This module extracts set of links and text from a fetched page within a page repository as shown in fig 1.9. Sometimes parsing also includes stemming of words, removal of stop words / common words, removal of HTML tags and Java scripts.

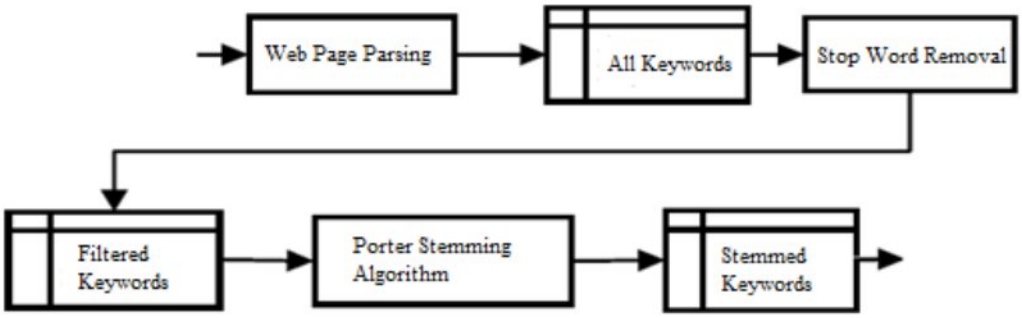

Fig 1.9: Parsing Module of web crawler

iv. **Fetch module**: This module uses the http protocol to access the web page using the given URL.

v. **Duplicate Elimination module**: It determines whether an extracted link or filtered link has already been passed to the URL frontier or has been recently fetched by the crawler. This module is also used for assigned a priority to the link based on which it is eventually removed from the frontier for fetching.

Initially crawler starts crawling from the seed URLs, fetch the web page content corresponding to those URLs, and links are extracted from the parsed page and stored in the frontier queue. Furthermore, crawler crawls by taking a URL from the frontier queue and fetching the web page corresponding to that URL, generally using the http protocol. The fetched page is temporarily stored into a page repository, where performs some operation on it. The stored page is parsed and links are extracted from the page. The link information and text are passed to the indexer where this information is used for ranking purpose. Furthermore, each links goes through a series of tests to determine whether the link should be added to the URL frontier or not.

The thread checks that the content of given web page is duplicated to another URL which had already been crawled. For detecting duplicate uses a simple fingerprint such as checksum and MD5 but more sophisticated test would be use shingles instead of fingerprints. In URL filter, there is a program which determines whether an extracted URL should be excluded from storing in frontier queue. Robot Exclusion Protocol (REP) [3] or simply robot.txt is a standard to communicate with web robots or web crawler which determines which portion of the web-site off-limits to the crawling.

In duplicate URL elimination, the URLs are checked for duplicates if the URLs are already crawled or already exist the frontier, we do not add them in it. A priority is assigned to the URL while entering in the frontier queue with the help of which it is removed from the frontier queue for fetching by the crawler. There are some housekeeping tasks which are also performed by the dedicated program. It wakes up once every few seconds to log crawl progress statistics such as, frontier size, URLs crawled *etc*, decide when to checkpoint the crawl or terminate the crawl. In check pointing process, a snapshot of the state of crawler is committed to disk. If a catastrophic failure occurs, then after that the crawl is resumed from the most recent checkpoint.

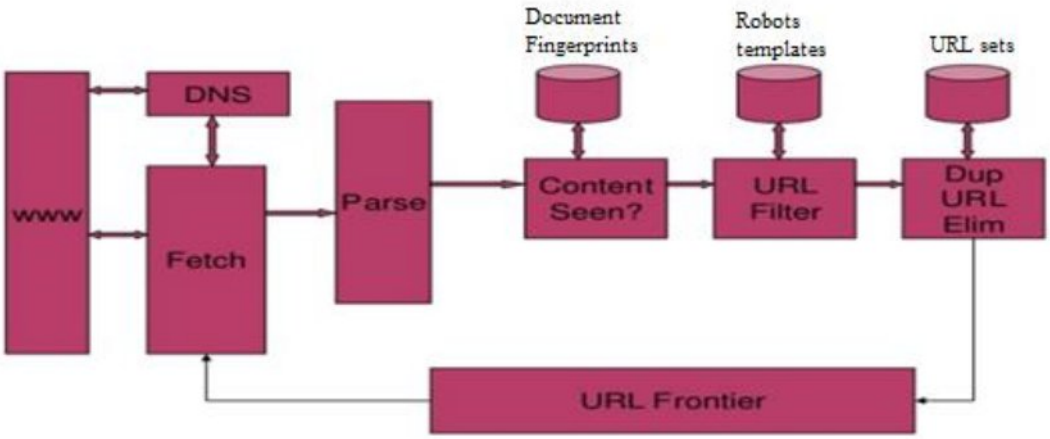

Fig 1.10: Architecture of Crawler [4]

Table 1.1 List of open source web crawler

S.No.	Crawler Name	Language Used	Platform	Description
1	JoBo	Java	Cross-platform	Basically it is a web spider which is a program to download complete websites. The advantage to other download tools is that it can use cookies for session handling.
2	WebEater	Java	Cross-platform	WebEater is a program written in Java for website retrieval and offline viewing of websites.
3	Ccrawler	C#	Windows	Ccrawler contains a simple extension of web content categorizer, which can separate the web pages based on their content. It was built in C# with Dotnet Framework.

4	Distributed Web Crawler	C, Java, Python	Cross-platform	An open source web crawler that uses Hadoop and map reduce.
5	iCrawler	Java	Cross-platform	iCrawler is an extensible crawler which will support adding any feature to it such as multitasking.
6	crawler4j	Java	Cross-platform	Crawler4j is an open source web crawler for Java which provides a simple interface for crawling the Web. Using it, we can setup a multi threaded web crawler.
7	Scrapy	Python	Cross-platform	Scrapy is an open source and collaborative framework for extracting the data from websites. In a fast, simple, and extensible way.
8	Nutch	Java	Cross-platform	Nutch is a well matured, production ready web crawler. It enables fine grained configuration, relying on Apache Hadoop data structure, which are great for batch processing.
9	Heritrix	Java	Linux	Heritrix is the Internet Archive's open source, archival-quality, extensible, web-scale web crawler.
10	GNU Wget	C	Linux	GNU Wget is a free software package for retrieving files using HTTP, HTTPS and FTP. It is a non-interactive command line tool, so it can easily be called from scripts, cron jobs, terminals without X-supports *etc*.
11	HTTrack	C/C++	Cross-platform	HTTrack is a free and easy-to-use offline browser utility. It downloads the whole website to a local directory, getting images, HTML, building recursively all directories and other files from the server to your computer.

1.4.5 Challenges in front of web crawler

There are some challenges occur in front of designing the web crawler because of the two factors [5]:

1. The size of the WWW is huge. There are some studies which indicate that after every 9-12 month the size of web gets doubled.

2. Web pages are updated very frequently. If we consider small change then about 40% of all web pages changes weekly. And if we consider large change then about 7% of the web pages changes weekly.

1.4.6 Crawling Policies

The web crawler's behavior is totally depends on the policies adopted by the crawler. These policies are as follows:

- Selection Policy: According to this policy, It must be decided by the crawler whether there is a requirement to download particular page or not.
- Re-visit Policy: In this policy, crawler must decide the rate for re-visit a particular web page.
- Politeness Policy: According to this policy, crawler knows that how to avoid overloading a web server.
- Parallelization policy: In this policy, crawler must know that how to coordinate different threads for crawling.

1.5 Types of web crawler

There are different types of web crawler based on different strategies employed in Web crawling. These are as follows [6]:

1.5.1 Generic web Crawler

Generic Web Crawler [1, 7] crawl the documents and links belonging to a variation of topics *i.e*. no restriction for crawling. Moreover, it usually stores each page individually and ignores the content relationships among pages. [8].

1.5.2 Focused web Crawler

Focused web crawler [9] crawls only those pages which are related to a particular domain *i.e.* focus on a particular topic and gathers documents which are relevant and specific to the particular topic, due to this nature of crawler it is also known as a Topic

crawler. This crawler is economically realistic in terms of network resources, hardware, decreases the network traffic and downloads.

Issues related with focused crawler

- **Finding Relevant Content:** In focused crawler finding the relevant content is the main issue. There is a possibility to download large number of irrelevant pages.
- **Maintaining the freshness of Index:** Many HTML pages consist of information that gets updated on daily basis. The crawler has to download pages and updates them into the database to provide up-to-date information to the user. Maintaining the latest version of web page within the index is one of the main issues.

1.5.3 Distributed Crawler

Distributed computing technique is used by the distributed crawler. In order to cover the entire web, many crawlers distribute the process of web crawling. A central server manages the synchronization of the nodes, as it is geographically distributed and communication between them. PageRank algorithm is used for increase the quality of search and efficiency of the crawler. The most important benefit of distributed web crawler is that it is robust against system crashes and other events.

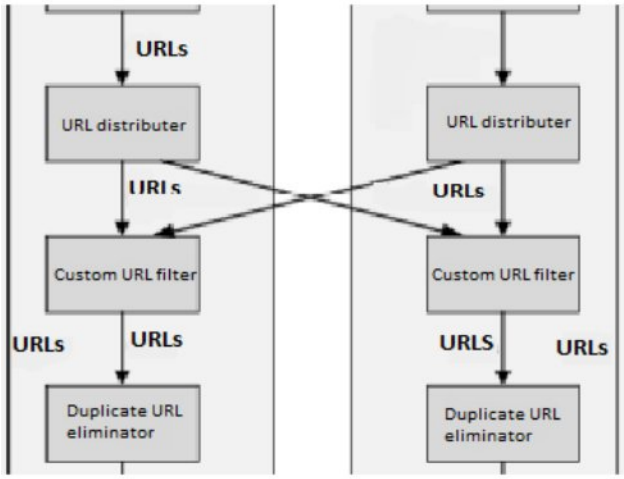

Fig 1.11: URL splitting in Distributed Crawler[5]

Distributed crawling is used for increasing the throughput. The throughput is increase because each crawl threads are run on a different machine or node. Each node is responsible for certain hosts. There are various policies on which hosts can be partitioned among various hosts. One of the partition policies for hosts is based on geographical location. A web page generally contains number of links to a page of

same site, some URLs whose responsibility falls under other nodes have to be forwarded to the respective nodes but most of the extracted URLs fall under the responsibility of the same node. We have a URL splitter which takes care of selection of URLs for particular node before the Duplicate Elimination module. Some examples of distributed web crawlers are UbiCrawler [10] and Merchator [5, 11].

1.5.4 Incremental Crawler

Incremental crawler is a traditional crawler which periodically replaces the old documents with the newly downloaded documents. This crawler incrementally refreshes the existing collection of pages by visiting them repeatedly based upon the estimate of how often pages change. The benefits of the incremental crawler are: it saves the network bandwidth, data enrichment is achieved, problem of the freshness of the pages is resolved and valuable date is provided to the user. Example of Incremental crawler is WebFountain [12].

1.5.5 Parallel Crawler

In parallel crawler, multiple crawling processes are run parallely. Crawl the significant portion of web or whole web becomes difficult using a single sequential crawler. A parallel crawler consists of multiple crawling Processes called as C-procs which can run on workstations network in parallel.

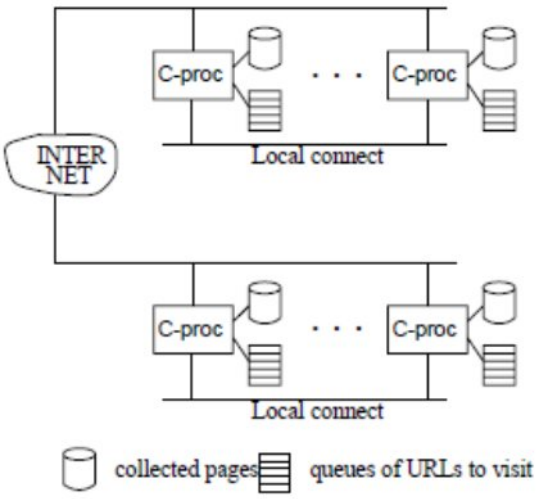

Fig 1.12: Architecture of parallel crawler[13]

C-procs works simultaneously to retrieve pages from the WWW and add those pages into central page repository of the search engine. The two main factors on which parallel crawler depend are page selection and page freshness. Parallel crawling increases the speed of downloading contents and can be used for either distributed network or local network. Example of parallel crawler is Cho and Molina's crawler [13].

Parallel crawlers are more challenging to operate. It has many important advantages as compared to single-process crawlers:

- Network-load reduction: In addition to distributing load, a parallel crawler may actually reduce the network load.
- Network-load dispersion: Multiple crawling processes of a parallel crawler may run at geographically distant locations, each downloading geographically-adjacent pages.
- Scalability: As the size of web is huge, it may be crucial to run a parallel crawler to achieve the required download rate in certain cases.

1.6 Redundancy in web crawler

The speed of information publishing in WWW is unprecedented due to which the internet is expanding in rocketing speed. According to the internetlivestats.com, there are about 1 billon websites on the WWW as on 01 April, 2016. This huge web contains billions of web pages characterized by their URLs.

The previous two studies, one from Digital, based on 30 million pages collected by AltaVista search engine during 1996 and another from Stanford, based on 26 million pages collected by the Google search engine during 1997, point out the large number of duplicate pages on the web. The digital study report showed that about 30% of the whole contents are duplicated and the Stanford study showed that about 36% contents of the web is duplicated. Both studies identified mirroring as the systematic replication of contents over a pair of hosts.

When collecting the web pages, a web crawler explores the web recursively by extracting all the linked URLs from a web page and put them into an unvisited URL list, which is known as URL frontier, so as to visit them later. However, the duplicate

URLs collected may lead to the result that quite a lot of pages may be visited and analyzed repeatedly. To visit the WWW efficiently, a web crawler must be equipped with an URL filtering module that helps in removing duplicate URL [11].

There are two ways by which duplication problem occurs [14, 36]: First, there are documents which are found in almost identical incarnation because they are:

- Combined with other source material to form larger documents.
- Same document with different formatting.
- Different versions of the same document.

Second, documents which are found in multiple places in identical form. Some examples are:

- Legal documents
- Documents stored in several mirror sites.
- Online documentation for popular programs
- FAQ (Frequently asked Questions)

We can get duplicate URLs from the web page in different ways. First, suppose we find a URL within a page which is already been fetched by the crawler. Second, suppose a different URL having the contents of another URL which is already been fetched by the crawler. We can say that two URLs having the same content. Third, suppose a URL having the content which is not identical to the contents of another URL but having similar contents. This type of documents called as Near-Duplicate document. The causes of Near-duplicate web pages are: exact replica of original site, mirrored site, versioned site and plagiarized documents. The Near-Duplicate detection plays a vital role in plagiarism detection, spam detection, focused web crawling scenarios and copyright violation.

Due to high rate of duplication in web document the need for detection and near-duplicate detection is increase in different applications like crawling, ranking, clustering, and archiving caching [15, 25]. The removal of duplicate URLs saves the bandwidth of the network, reduces the costs of storage and improves the quality of the search indexes. It can also reduce the load on the remote host which is serving such web pages. We can find the duplicate URLs with the help of finding duplicate and near-duplicate contents detection techniques.

1.6.1 Duplicate Documents

Two documents are said to be duplicate if both the contents are bitwise identical to each other. Two documents are duplicated if and only if the different URLs referenced to the same web page. This is also known as Different URL with similar content (DUST) [16, 26].

1.6.2 Near-Duplicate Documents

Documents which have small dissimilarity and are identical to a remarkable extent but are not identified as being exact duplicate of each other [15]. These documents are not bitwise identical to each other.

Some examples of the near-duplicate documents are as follows:

- Files with the same content but different file type *e.g.* Microsoft word and PDF version of the same file.
- File with the same content but different formatting *e.g.* the documents might contain the same text, but dissimilar fonts, bold type or italics.
- Files with a few different words *e.g.* these types of documents are widespread form of near-duplicate.

Instead of search engine, there are some other applications which are benefited by identification of near-duplicates. Some of the following applications are [17]:

- The detection of the near-duplicate web page helps in identification on spam, enhanced quality and diversity of the query results, focused crawling *etc*.
- Web mining applications which requires near duplicate identification are detecting plagiarism, community mining in a social network site, document clustering, collaborative filtering, detection of replicated web collections, discovering large dense graphs, and.
- Data cleaning and Data integration in database systems require identification and elimination of NDD.
- Digital libraries and electronic published collections of news achieves require NDD removal.

1.7 Duplicate detection system for web crawler

Web crawler has the ability to store the content of the web pages and provide a content identifier (CID) as shown in fig. 1.13. CID is a unique identifier for each document which is a unique function of the physical storage location. Crawler first tries to find the CID of a document. If the crawler finds the CID of a document in the document store, the CID is fetched from the document store. If document store does not support the CID attribute for a document, the document is fetched by the crawler, filter the document to produce a hash function, and store the document to an index if the hash function is not present in the history table. The crawler determines whether the CID is present in the history table or not. If the CID is not present in the history table, the full document is retrieved and indexed. If the CID is present in the history table which indicates that the document has already been indexed, the new URL is placed in the history file but the document itself is not retrieved from the document store and the document is not filtered again to obtain a CID. The data structure of CID is an extension of a globally unique ID (GUID) which is a 16-byte number; the CID comprises a 16-byte GUID plus an additional 6-byte number [18].

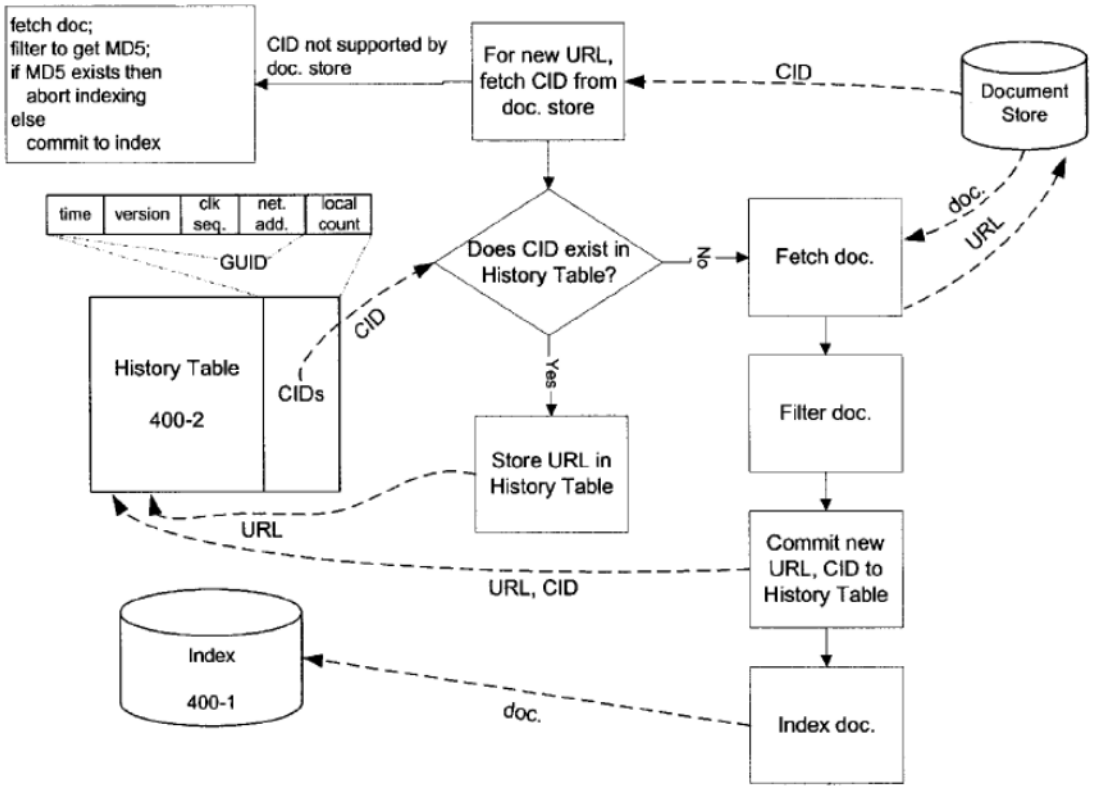

Fig 1.13: System for duplicate detection [19]

Chapter 2

Literature Survey

2.1 Types of duplicate contents for web crawler

The elimination of duplicate URLs from the frontier queue of the crawler plays vital role because it increase the efficiency of the crawler. The removal of duplicate URLs saves the bandwidth of the network, reduces the costs of storage and improves the quality of the search indexes. It can also reduce the load on the remote host which is serving such web pages. When crawl the web using a normal link tracer, it is very common to go through a link more than once. If this link is crawled multiple times, a duplicate of this page will be added into our page repository. Unless the content has substantially changed since the last crawl, we will have various copies of the same web content in our index.

Duplicate pages in a search index can create redundancy in the search results. The goal of a search result is to return both relevant and diverse documents, allowing users to decide the optimal resolution for a query. Without Deduplication, the top-k results returned for a user's query will likely contain duplicate content. In the extreme, all the k results will be copies of the same page. This creates a bad user experience where, as the crawler scales out, the duplicate likelihood increases. In fact, Google's Matt [37] Cutts believes that up to 20% of the web content is duplicated. We depicted different types of duplicate web content for web crawler as a funnel in fig 2.1.

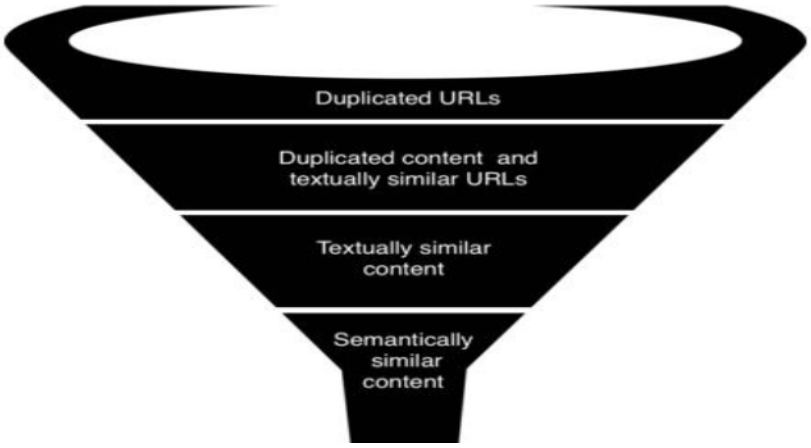

Fig 2.1: Different types of duplicate content for web crawler

At the top the aim is to detect a large percentage of obvious, exact duplicate content with minimal overhead. Moving down the funnel, the techniques for near-duplicate detection becomes more robust to small differences in page content at the expense of larger overheads.
Let's explain each step one by one.

2.1.1 Duplicate URLs

The goal at the top of the funnel is to discover and remove a large number of obvious duplicate URLs We can find the duplicate URLs with the help of hash map of previously crawled URLs. The memory requirement for such a map will increase as the number of unique URLs that has been crawled will become huge. And to avoid this memory explosion, while maintaining fast duplicate lookups, a bloom filter [20] can be used. A bloom filter is a data structure designed to tell, memory efficiently and rapidly, whether an element is present in a set.

2.1.2 Textually similar URLs

Frequently the URLs on a given page are not exactly the same. There exist some parameters embedded within a URL that different it from others. For example, www.news.google.com and www.google.com/news both refer to the same web page. Ziv Bar-Yossef *et al.* [16] suggested a technique for identifying the different URLs having the similar text (DUST), known as DUSTBUSTER algorithm. It used previously crawled logs to learn canonical representation of the URLs for a given site and built a set of substring and parameter substitution rules to canonicalize uncrawled links.

2.1.3 Duplicate content

With the help of the first two categories mentioned above duplicate content can be identified for a particular domain. To find the duplicate content across another domain, it required to crawl the web link and its corresponding HTML page has been retrieved. Likewise the duplicated URLs, once a page has been placed in the page

repository, its duplicated content can be effectively detected with another bloom filter [20].

2.1.4 Textually similar content

All the above methods can easily identify the exact duplicate content. One of the most tedious task is to find the near-duplicate web content where the text is not bitwise identical, but present with slight changes. To identify such content, a computational intensive method named locality sensitive hashing can be applied for the identification.

The selection of best strategy in the above mentioned four methods for Deduplication of web content will affect the overall performance of the web crawler. Till date various techniques are proposed in the field of web crawler. Works in this area of Deduplication [7] of web content started in the year 1982. As the year passed by, the industry starting realizing the benefits of eliminating the duplicate URLs new approaches came up. Taking into consideration various ideas put forth by the industry and academicians, major focus of this literature is to provide review of all the different approaches used for eliminating the duplicate URLs by providing a chronological sequence of all the research activities done in this area starting from year 1994 till date.

2.2 Near-Duplicate Content Detection

With the increase amount of data and the need to integrate data from multiple data sources, a challenging issue is to find near-duplicate web pages efficiently [34]. We can easily find the exact duplicate of each other by standard checksumming techniques, but the identification of Near-Duplicate Document (NDD) is more difficult [17]. Duplicate and Near Duplicate web pages are creating large problem for search engines. These web pages increase the space needed to store the index, either slow down or increase the cost of saving results and annoy the users. Deduplication saves network bandwidth, reduces costs and improves the quality of the search indexes. It also can reduce the load on the remote host that is serving such web pages [19].

2.3 Approaches for NDD detection for web content

For finding the near-duplicate document for the web content is a well known open problem, and a number of approaches have already been proposed for the same. Fig.2.2 depicts that in general, we broadly categorize the different approaches for near-duplicate document detection of web content into two main categories, conventional and unconventional approach. The conventional approaches include keyword based approach, URL based approach, cluster based approach, shingling approach and fingerprint approach. These approaches are called conventional approach because these are generally accepted. The approaches other then fingerprint approach are just a concept *i.e*. not used in any web crawler. The unconventional approach includes locality sensitive hashing because this technique is recently used in web crawler to detected near-duplicate detection.

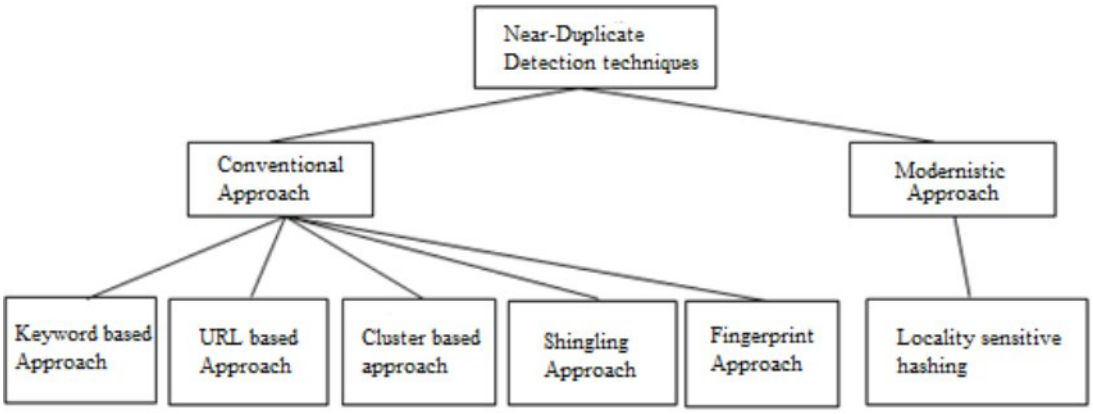

Fig 2.2 Different approaches for near-duplicate document detection

2.3.1 Fingerprint Approach

Udi Manber [27] proposed an approach for a system known as SIF, which can be used to uncover similar files from a bulky file system. The key objective of the work was to recognize the content similar to the some pre-existing source. The technique given by author has differentiated itself from fingerprint and checksum approaches. In this scenario two files were considered similar even if the similarity score was less than 25%. In the proposed approach the author found all probable substrings of a definite length and signature of each substring followed by making a subset of hash based signatures. Here, the two substrings with same fingerprint were said to be

identical and two identical substrings will generate the same fingerprints, regardless of their position in the text.

Charikar's Simhash [28] proposed a technique to identify near duplicate documents which mapped a high dimensional vector into a small sized fingerprint. Sketch algorithm was used to estimate similarity which constructs a function for producing compact sketches of the document in a collection. Here, a web page was converted into a set of attributes where each attribute was associated with its weight. Further, attributes were fixed using the standard information retrieval (IR) techniques *viz.* stemming, stop-word removal, case folding, phase detection, and tokenization [22]. A high dimensional vector was formed by a set of weighted attributes. Each dimension was associated with unique features in all documents. After the above step, high dimensional vector was changed into f-bit fingerprint by Simhash's algorithm. The algorithm had been generated similar hash values, small hamming distances, for the documents having single byte difference, but in cryptographic hash function like MD5 or SHA1 generates different hash values for the documents having single bit difference. In this technique, 64-bit fingerprint is sufficient to represent a repository of 8B web pages.

S. Brin *et al*. [21] proposed a system for registering documents and detecting the duplicates (either complete or partial duplicates). Here, the main idea was to maintain a server where the registered original documents detected for the duplication. The proposed approach focused on a specific class of operational test known as ordinary operation tests (OOT) [29]. The system also explained a prototype related to the implementation of the services, called copy protection system (COPS).

At the time of developing a system for NDD detection for repositories having billions of pages, Gurmeet S. Manku *et al.* [23] provided two seminal contributions. Firstly, confirming that the fingerprinting technique which was generated by Charikar [28] was appropriate for finding the near-duplicate. Secondly, technique was introduced for identifying the given fingerprint that differs from existing f-bit fingerprint in at most k-bit positions. The proposed approaches were helpful for both multiple fingerprints (batch queries) and single fingerprints (online queries) [30].

In a related work, Nevin [55] focused on fingerprinting technique which was effective on large environment and can also identify those documents which had

similarity equal to or less than 5% [57]. In this approach the documents is divided into sub-sequences of characters and then generating a fingerprint based on the hash-value of sub-sequences. In full fingerprinting, similarity between two documents is measured by counting the number of common sub-sequences in the whole fingerprints. Because of the size of the fingerprints generated, the full process can become impractical. To reduce the size of the fingerprints, a subset of the substrings is selected. Author compared the match percentage of selective fingerprints against full fingerprints to demonstrate the selective fingerprints normally perform better than the full fingerprints.

Narayanan Shivakumar *et al*. [31] proposed an approach to detect the duplicates, in which frequencies of the words occurring in the new document were compared against the stored documents. Authors divided the techniques of duplicate detection into two groups' *viz.* registration, and signature based. Within a scheme of signature based, a fingerprint was added along the document and fingerprint was used to track the document owner. This scheme was having two main flaws *viz.* (a) the signature could be detached robotically, which leads to untraceable document, and (b) partial overlapping couldn't be detected from this. Overlap checking involved many arbitrary problems in the database of registration, which was costly. To sustain their claim compared their prototype Stanford Copy Analysis Mechanism (SCAM) with the Copy Protection System (COPS) on 1233 * 1233 Netnews article pair and showed that SCAM was performed better than COPS in detection instance of plagiarism. Although noted that COPS reports less false positive than SCAM, false positive indicates that a given condition is present when it is not. SCAM also performed better in document overlap detection while comparing with traditional vector based Information retrieval scheme on the same 1233 Netnews article.

Midhun Mathew *et al*. [32] proposed an algorithm which was based on term document weight (TDW) matrix. The algorithm comprised three phases *viz.* First phase was rendering phase in which it collect input from web pages and threshold values, second phase was about filtering where positional filtering and prefix filtering were used to minimize the size of records, and third phase was about verification where calculation of the similarity score of web pages followed by returning a best possible set of Near-Duplicate web pages was performed. This approach not only works on the content of web pages but also on the context of web pages. The entire

preprocessing and weighting scheme was done with the global ordering by TDW matrix in the rendering phase. The filtering techniques such as positioning [28] and prefix filtering were used to minimize the size of the records that could reduce the number of comparisons among the records; this was done in filtering phase. In the last verification phase, researchers checked the similarity based on the threshold value and finally evaluated the best possible number of Near-Duplicate records. Authors have used two standard benchmark measures to evaluate the scalability and accuracy of their algorithms, namely, precision and recall.

Arun PR *et al*. [34] proposed a method which deals with duplicate and Near-Duplicate web page detection by using an extended term document weighting (TDW) scheme, sentence level features and Simhash's technique [23]. The proposed approach used the term document weighting (TDW) scheme for finding the term importance and Simhash technique for fast document comparison. The near duplicate detection algorithm could be effectively executed prior to the indexing of downloaded content. The proposed approach incorporated four phases as preprocessing, word mapping, feature extraction, and feature comparison. The experimental results showed that the execution time for the TDW based approach was increasing with the size of the repository where the extended TDW with Simhash's approach haven't gradually increased with the number of pages.

Ernesto Di Iorio *et al*.[35] proposed a technique based on a pair of signatures to detect similar web pages. Both signatures were low dimension vectors in order to lower the computational expense for comparing pairs of documents. The document should not only compare with their content, but also needed to compare with their hypertext context by using Hypertext Map signature. A random projection of the bag-of-words vector representing the page content was used to obtain the first signature. To reduce the dimensionality of the document random projection was used instead of a dictionary based representation, which lowers the computational expenses for comparing two documents. A recursive equation, which utilized the connectivity between the web pages to code the context of each page, was used to obtain second signature known as hyperlink map. In random projection signature, only the content of the page was considered. The experimental results showed that by using random projection signatures and hierarchical map signature together, a high precision and recall can be achieved for finding the duplicate and near-duplicate.

2.3.2 URL Based Approach

For a document with two or more versions, this is not mandatory to have all the versions as identical. Variations may exist in terms of date, counter, and advertisement; these terms could be irrelevant in the context of content. The WWW is full of different URLs with similar text (DUST). Suppose A and B are two URLs, they called as DUST, if and only if their corresponding contents are identical. For example, the URLs http://abc.com/ and http://abc.com/index.jsp return similar content, abc.com/default.html and abc.com/index.html will return the same result.

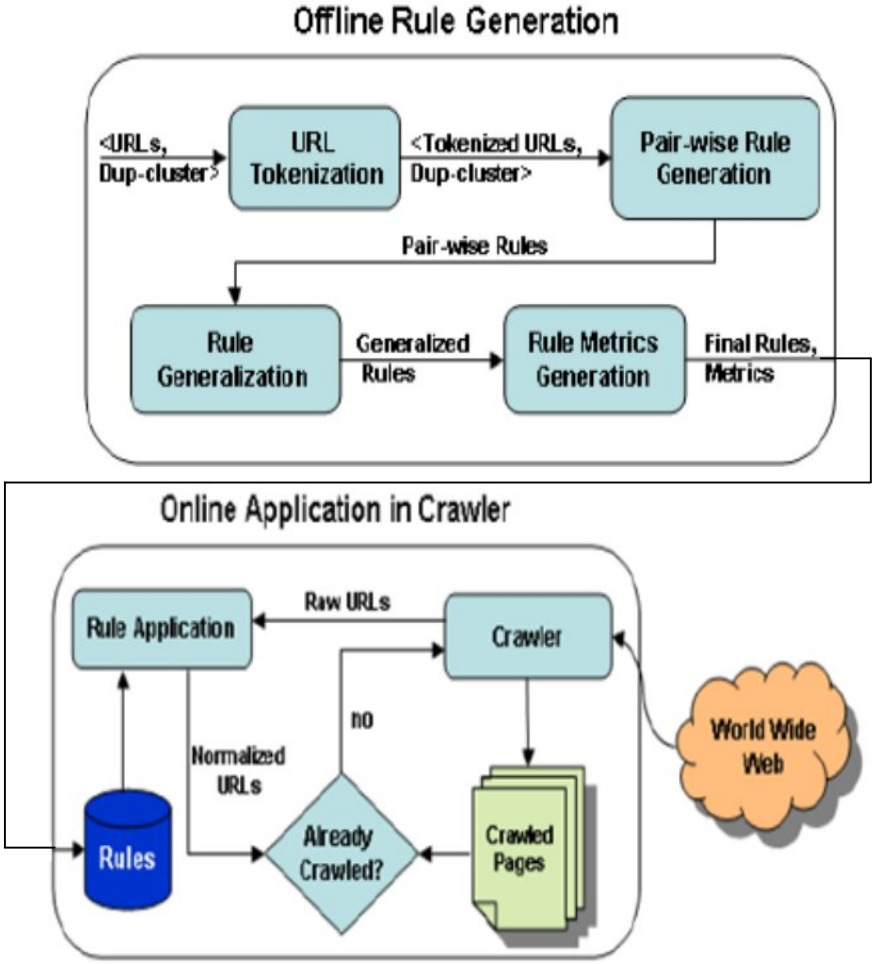

Fig 2.3 Diagram of online Rule Generation and offline Rule Application [36]

One of the most common sources for DUST could be mirroring of websites or web content. Ziv Bar-Yossef *et al.* [16] proposed a new algorithm, Known as DustBuster, for removing DUST. DustBuster found set of rules that convert a given URL into other URLs having the identical content. It mined DUST from web server logs or previous crawl logs. A list of URLs was maintained from many sources,

including web server logs or previous crawled logs. The algorithm, DustBuster, generated set of rules from the previously generated URLs list. The set of rules was verified by sampling a small number of actual web pages. It has been experimentally showed that DustBuster reduced a crawling effort by 26% as it discovered 47% of the DUST. Amit Agarwal *et al* [24] proposed a cluster of techniques for mining the rules from the URLs and then used those mined rules for near-duplicate detection without fetching the content of the page explicitly. The technique involved mining of crawled logs and utilizing similar clusters to extricate rules. There should be at least one rule for each cluster and generalize the set of rules with the help of machine learning techniques [39].

The clusters of URLs with similar page content, such a cluster is known as a duplicate cluster or a dup-cluster. The first step in Offline Rule Generation is tokenizing the URLs into <key, value> tuple. These tuple along with the dup-cluster information are used for the Rule Generation. Pair-wise Rules are generated from selected URL pairs with in a dup-cluster and then the pair-wise Rules are generalized. Generalization not only reduces the number of Rules but also give Rules which can efficiently normalize unseen URLs. Applications such as crawlers, while crawling obtain a set of new URLs to crawl. These URLs are normalized using Rules generated from o²ine processing. Normalized URLs are compared with already crawled URLs to find duplicates. This process of de-duplication avoids the overhead of crawling duplicate documents. As crawlers and other real-time applications have resource constraints, online processing has to generate a small set of Rules which can achieve maximum reduction in duplicates. The main advantage of using URL based approach is that we can find the duplicate without explicitly fetching the content of the web pages.

Yingjun Wu *et al.* [35] proposed a new variation of bloom filter known as hierarchical bloom filter (HBF) to reduce the false positive rate by taking advantage of the working principle of web spider as well as the hierarchical structure of URLs. HBF has reduced the false positive rate with the help of multiple level bloom filters. However, a good balance between the false positive rate and space occupation is hard to find out. HBF was a modified URL filtering method. The proposed approach has achieved the aim of reducing the false positive rate without incrementing the memory usage, and that was because of URL dictionary hierarchy. According to the

experimental analysis and evaluation, this new URL filtering method could significantly reduced the space complexity by at most 50% theoretically while the probability of false positive remains the same.

Krishna Bharat *et al.* [41] proposed an approach to identify a mirrored host from the syntactic analysis of the URL string. It required content analysis and retrieval of web page only for a less number of pages which identified both total & partial mirroring, and managed cases in which content was not bitwise identical [42]. The detection of mirror sites was very important to compensate the broken links and to avoid redundant fetching. The approach was spotted diverse degrees of mirroring and it was important to distinguish between content and structure to classify them. It defined the structure as the set of legitimate paths relative to the given host. While considering the content, if two pages were similar in context of bytes then it can be named as content identical pages. While considering mirroring, without changing the content of pages, a page usually get changes at the byte level. If the two hosts have the same set of paths then those hosts were said to be structurally identical hosts. If the two web pages were identical in term of content after the normalization then those pages were said to be content equivalent pages. Two pages were highly similar if and only if the pages changed in content, *i.e.* due to dynamic content or banner advertisement The authors have described the high similarity based on the edit distance or some other appropriate measure and used the similarity measurement technique, which had already been used in [31], with an appropriate threshold. Two pages were said to be related if and only if the pages were semantically similar but change significantly at the syntactic level. The application of this approach was to improve reliability and caching behavior, to reduce redundant fetching in crawler and to detect mirrors in web proxies.

2.3.3 Shingling Approach

Andrei Z. Broder *et al.* [31] introduced a systematic technique to know the syntactic similarity within the files and also applied the syntactic similarity within the documents on the web. The author used the statistical concepts related to containment and resemblance for finding the similarity between the documents. The containment score between the two documents can be defined as the degree at which one document

contains within another document and its value varies between 0-1. The score near to 1 indicates about containment of one document into another. Similarly, the resemblance score can be defined as the degree of duplication between two documents. For this also the score varies between 0-1 and the score near to 0 indicates that the two documents are not near-duplicate to each other. The documents were analyzed lexically into a canonical sequence of tokens. The minor details such as capitalization, HTML commands and formatting was ignored by that canonical form. Shingle can be defined as a contiguous subsequence contained in document Doc [32]. Each document was associated with a set of the subsequence of tokens having size w words S(Doc, w). A 40-bit fingerprint function was used and enhanced it to behave as a random polynomial. This fingerprint function was based on Rabin fingerprint [38]. The documents were divided into tokens or shingles and then started the computation on each shingle separately and then merged the results. Suppose a shingle repeated in many documents, *i.e*. 1000 document, this will create a performance issue. This approach allowed users to find documents on the web which were syntactically similar [40] and further allowed the crawler to eliminate duplication based on the similarity between two documents.

2.3.4 Keyword Based Approach

V. A. Narayan *et al*. [37] proposed an approach for detection of Near-Duplicate document in which extraction of the keywords was performed from the crawled web pages followed by finding the similarity score, based on the keywords extracted from the web pages, between two pages and set some threshold value. The documents could be considered as near duplicate if their similarity score lied above the threshold value. Near-duplicate detection was performed only on the extracted keywords instead of whole content of web page[56]. Parsing was used to extract the keywords from the web pages. It included stemming of the remaining words, stop words, html tag, and script tags. To ease the process of near-duplicate detection, the extracted keywords and their count was stored efficiently in the table to keep the search space minimized. The main advantage of this approach was that it reduced the storage space that improved the quality of search engine.

Jack G. Conrad *et al.* [43] purposed an approach which can mitigate the effect of near-duplicate or duplicate content in the search results. Most of the topical work was

related to reduce the complexity of document comparison by converting a document with signature. A technique was used to produce a signature of important feature within a document and found these features using collection statistics. The term's inverse document frequency has been used in collection statistics. The term's inverse document frequency (tf-IDF) [51] is a numerical statistic which is intended to reflect the importance of a word within a document in a collection. The new versions of the document was frequently arrived in the page repository and the collection of documents was always updated with recent documents were two most important challenges in front of this approach. If a term appears less than five times across the collections then that term will not be considered for table membership. With the help of these restrictions many attribute have been filtered which can lead to typographical errors, misspellings and others.

Nilakshi Joshi *et al*. [44] proposed an algorithm known as a NDupDet algorithm for near-duplicate web page detection to increase search productiveness and storage efficiency of search engine. The algorithm consisted of three major steps *viz*. Firstly, calculate the frequency of each keyword in a web page. Secondly, these keywords were sorted according to the descending order of the frequencies. Third, select top N Keywords from this list. The value of N was set to 5 and 10. Whenever, a new page arrived and needs to be stored in the repository, its similarity with already existing web pages in the repository needed to be compared.

Chuan Xiao *et al*. [33] introduced the exact similarity join algorithms that were used in NDD detection. Researchers have proposed a concept that utilized the token ordering within a record, known as positional filtering, which further leads to find the upper limit of similarity score. Authors also proposed a novel filtering technique by utilizing the ordering information and incorporated with the existing methods which reduced the candidate size drastically, which leads better efficiency. Researchers were looking for an efficient approach to find out the pair of records whose similarity score was above a given threshold. With the help of similarity function researchers could easily found that whether two objects were near-duplicate or not. The similarity function calculated the extent of likeness between the two documents and the function will returned a value between 0-1. Similarity function could be chosen on the basis of the application domain. Some extensively used similarity functions were Cosine similarity[59], Overlap similarity[60], and Jaccard similarity[54].

2.3.5 Cluster based approach

Junghoo Cho *et al.* [40] described the proficient collection and identification of the replicated documents. The main obstacle in front of identifying duplicates was the input size of data sets which was about millions of web pages containing thousand of gigabytes of textual data. Duplicated information can be used to improve a search engine and presented two real-life case studies to support the statement. Identification of similar clusters within a given web graph and finding the similarity between different web pages based on the clusters. The main goal was to identify the mirrored collections on the WWW with the help of which a number of tasks such as crawling, ranking, web archiving and caching can be performed more effectively.

Lavanya Pamulaparty *et al.* [45] presented a framework named as extensible NDD Framework (XNDDF) having modules that helped to provide flexible solutions to duplicate detection as well as showing online and offline processing necessary by a search engine. Two algorithms have been introduced for near-duplicate detection and document clustering. The first algorithm used document local features with the help of similarity measures for NDD detection and weighted terms. This algorithm has taken the support of a classifier. The Second algorithm can be used, prior to NDD for document clustering followed by unsupervised probabilistic clustering approach [58]. The similar document clustering reduced the area of document comparison. The NDD algorithm was provided very proficient decisions by using computing discriminating function, but for that it was important to train the classifier. In order to compute discriminate function, domain experts were provided the training documents to the classifier.

Dennis Fetterly *et al.* [46] experimentally proved that the clusters of NDD were reasonably stable. If the two documents were near-duplicate to each other then after 10 weeks these were still near-duplicate. It was also confirmed by the Broder et al. [31] that the widespread near-duplication of web pages and found that about 28% of all web pages were duplicate of some pages and 22% are virtually identical out of remaining 72%. The documents from 20 large clusters were investigated and categorized into clusters. The exit rate of the document from the cluster were examined and found that the clusters were literally stable over time. Intermediate

sized clusters were generally the least stable. The practical importance of this work implied that there was no need to crawl the web page within the same cluster.

Ranjna Gupta *et al.* [47] proposed an architecture which introduced modules that run offline as well as online on the foundation of disfavored, and favored user queries to detect Near-Duplicates and duplicate documents so that relevant search results can be displayed to the users. The proposed architecture that works offline comprised of the following functional components: query clustering tool, favored query finder, and duplicate data detector. Once query clusters were formed, the subsequent step was to find a set of favored and disfavored [48, 56] user queries from each cluster. A query was said to be favored query that occupy a most important portion of the whole search request in a cluster and it was understood that their analogous search results have more duplicates in search results than disfavored queries.

Chapter 3
Problem Statement

After reviewing the literature of duplicate and near-duplicate detection techniques for the web crawler, it has been analyzed that the literature can be divided into two broad categories *viz.* conventional approach and modernistic approach. These categories can further be compared on the basis of parameters like time taken by the crawler to traverse a website, and number of duplicate document detected by the crawler. Although there exist a lot of literature for keyword based approach, URL based approach, cluster based approach, shingling approach, fingerprint approach, and locality sensitive hashing, but still there exist an opportunity to frame all the available information into a more structured way to provide a systematic analysis regarding the usage of the said techniques in specific domains.

Chapter 4
Methodology

The proposed work addresses the comparison of conventional and modernistic approaches for near-duplicate detection on the basis of time taken by the crawler to crawl the websites using different algorithms and number of duplicate documents detected by the crawler. The algorithms implemented on Nutch crawler are MD5Signature, TextProfileSignature and Simhash algorithms. The first two algorithms are comes under the category of conventional approach and Simhash algorithm is comes under the category of modernistic approach.

For implementing the comparative analysis between MD5Signature, TextProfileSignature, and Simhash algorithm following steps have been followed:

1. Installation of software which is required for running Nutch web crawler [49] on windows. These software are as follows:
   1.1. Cygwin 2.5.1
   1.2. Apache 8.0.35
   1.3. Apache Solr 2.1.1
   1.4. Nutch 1.8.0
2. Implementation of MD5Signature, TextProfileSignature and Simhash algorithm on Nutch crawler.
3. Select the seeds website *i.e.* Thapar.edu, gr8ambitionz.com, examrace.com, Stanford.edu, amazon.com. We have selected these websites on the basis of number of URLs in it. Therefore, the number of URLs within the websites ranging from 80 to 1184.
4. Compares the time taken by the Nutch crawler using different algorithms and seed URLs.
5. Compares the number of duplicate documents detected using above mentioned algorithms in the Nutch crawler.

Chapter 5

Experimental Results

This chapter focuses on implementation of two well known near-duplicate detection techniques which are fingerprint technique and locality sensitive hashing technique. These techniques are implemented on Nutch crawler along with cygwin and solr.

5.1 Tools used for implementing duplicate detection techniques

5.1.1. Cygwin

Cygwin [52] is a large collection of GNU and open source tools which provide functionality similar to Linux distribution on Windows. It is not a way to run native Linux apps on Windows. You must rebuild your application from source if you want it to run on windows. A DLL file (cygwin1.dll) which provides substantial POSIX API functionality. The Cygwin DLL currently works with all recent, commercial released x86 32 bit and 64 bit versions of windows, starting with Windows XP SP3. In our experiment we are using most recent version of the cygwin DLL is 2.5.1. Install it by running setup-x86_64.exe. The cygwin is shown in fig.4.1.

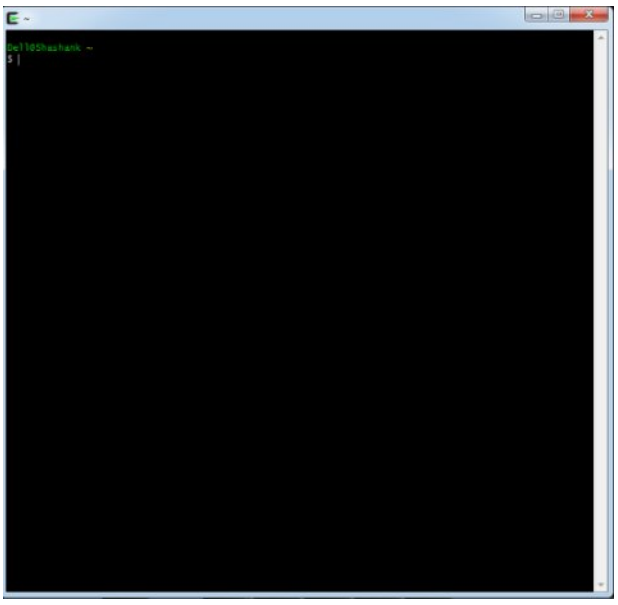

Fig. 5.1: Cygwin in Windows 7

5.1.2 Apache Tomcat

The Apache Tomcat [50] software is an open source implementation of the Java WebSocket technologies, Java Expression Language, Java Server Pages and Java Servlet. In this implementation latest version has been used which is version 8.0.35 of Apache Tomcat and using port no. 8080 which is the default port number for Tomcat. For successfully run the Apache Tomcat we need to set the environmental variable names as Catalina_Home.

Fig 5.2 Apache Tomcat Server using Port 8080

5.1.3 Apache Solr

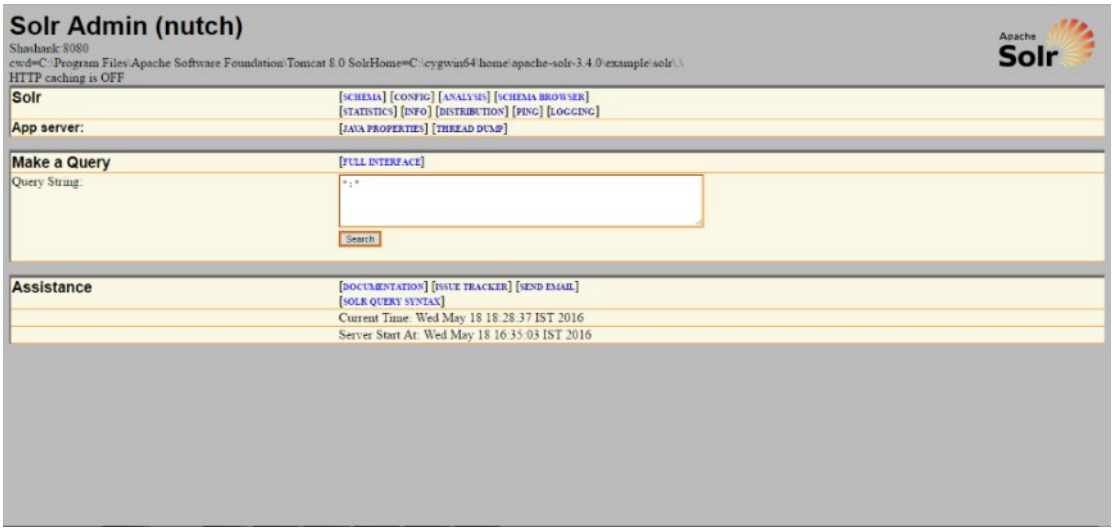

Fig 5.3 Apache Solr

Apache Solr [51] is an open source standalone enterprise search server with a REST-like API, which is written in Java, from the Apache Lucene project.

We can put documents in it (called “indexing”) via JSON, XML, CSV or binary over HTTP and query it via HTTP GET and receive JSON, XML, CSV or binary results. We can run the solr using Apache Tomcat server.

5.1.4 Apache Nutch

Apache Nutch is open source web crawler software which is scalable and highly extensible, written in Java, developed by Apache foundation. We are implemented Nutch by using Cygwin as depicted in fig 5.4.

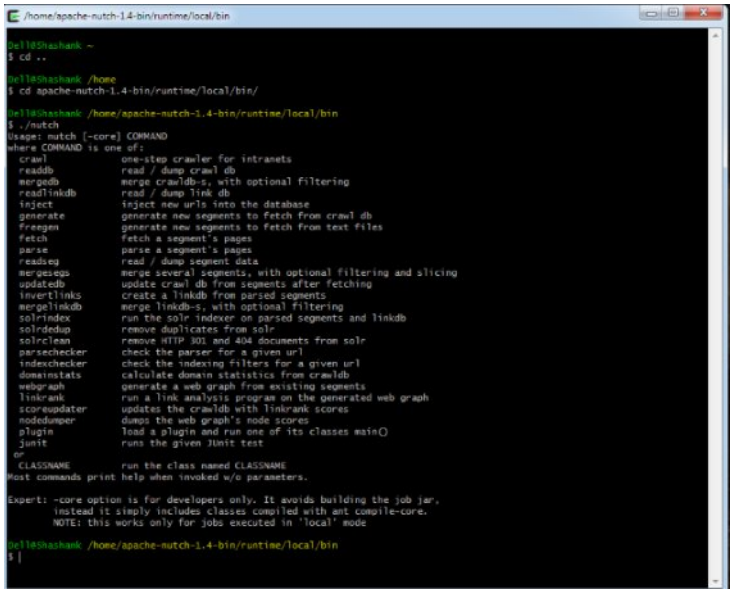

Fig. 5.4 : Implementation of Nutch using Cygwin

5.2. Implementation

There are two main techniques for duplicate detection in web crawler. These are fingerprinting techniques and locality sensitive hashing. The algorithms within these techniques are implementing on Apache Nutch 1.8.0.

5.2.1. Fingerprinting (Signature) technique

Preventing near-duplicate or duplicate web pages from entering into an index or tagging web pages with a signature for duplicate field collapsing can be efficiently achieved with a fuzzy or low collision hash function. The Signature can be implemented in several ways:

Table 5.1 Algorithm used under fingerprinting techniques

Methods	Description
MD5Signature	128 bit hash for exact duplicate detection.
Lookup3Signature	64 bit hash used for exact duplicate detection, much faster than MD5 and smaller to index.
TextProfileSignature	Fuzzy hashing implementation from Nutch for near-duplicate detection. It's tunable but works best on longer text.

5.2.2. Locality Sensitive Hashing

Locality sensitive hashing is a technique to reduce the dimensionality of high dimensional data. Locality sensitive hashing differs from cryptographic hash function such as MD5 signature and Lookup3 signature because it aims to maximize the probability of a "collision" for similar contents. Simhash algorithm is a very efficient variant of locality sensitive hashing in terms of speed and space used.

In our experiment we are considered five websites, each website has different number of URLs as depicted in table 5.2. We have found the URLs within a website using a tool "Website link count checker".

Table 5.2: Number of URLs corresponding to websites

websites	Number of URLs	Internal Links	External Links
Thapar.edu	196	177	19
gr8ambitionz.com	202	180	22
examrace.com	1184	1178	6
Stanford.edu	80	19	61
Amazon.com	138	73	65

The time taken by the Apache Nutch crawler to crawl the websites using algorithms such as MD5Signature as M, TextProfileSignature as T and Simhash algorithm as S as depicted in Table 5.3.

Table 5.3: Time taken by the crawler to crawl website using different algorithms

	M	T	S
Stanford	2.535556	2.640833	2.676944
Amazon	6.206389	6.2825	6.315833
Thapar	7.450278	7.506667	7.539167
Gr8ambition	10.19194	10.14861	10.0825
examrace	15.52306	15.08306	14.90472

The time taken by the Nutch crawler to crawl website www.stanford.com using MD5Signature as M, TextProfileSignature as T and Simhash algorithms as S as depicted in fig 5.1. As the site contains 80 URLs, out of 80 URLs there are 19 internal links and 61 external links. So MD5Signature algorithm works well as compared to other two algorithms.

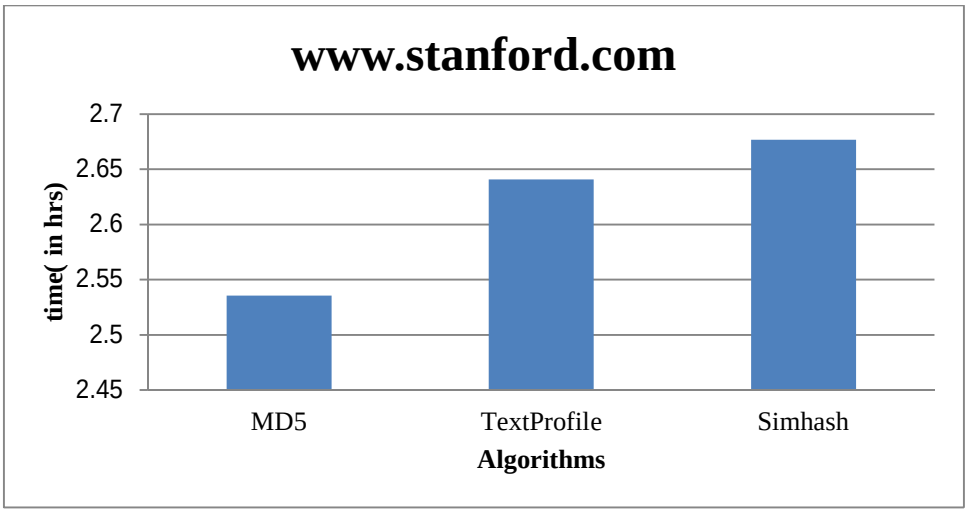

Fig 5.5 time (in hrs) taken by website (www.stanford.com)

The time taken by the Nutch crawler to crawl website www.amazon.com using MD5Signature as M, TextProfileSignature as T and Simhash algorithms as S as depicted in fig 5.6. As the site contains only 138 URLs, out of 138 URLs there are 73 internal links and 65 external links. So MD5Signature algorithm works well as compared to other two algorithms.

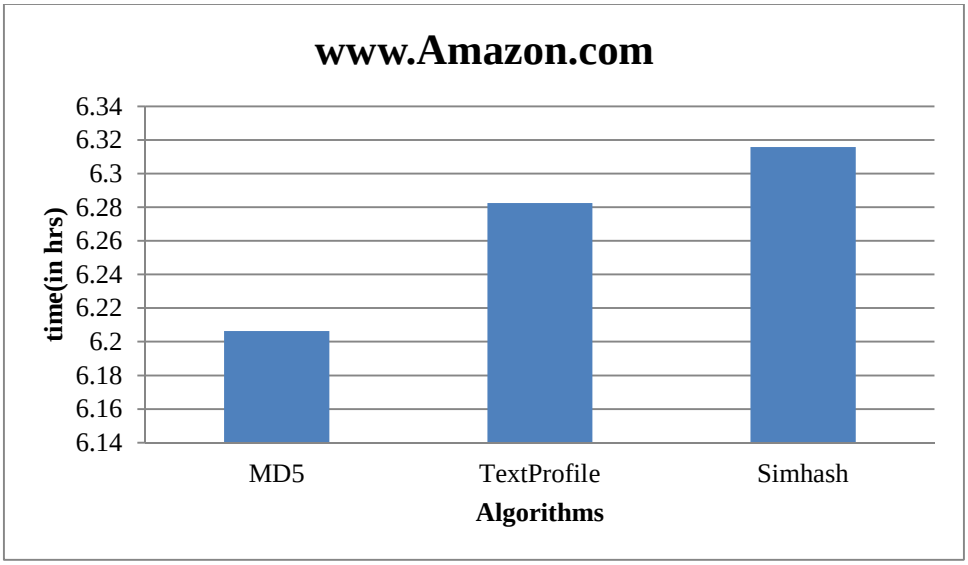

Fig 5.6: time (in hrs) taken by website (www.amazon.com)

The time taken by the Nutch crawler to crawl website www.thapar.edu using MD5Signature as M, TextProfileSignature as T and Simhash algorithms as S as depicted in fig 5.7. As the site contains only 196 URLs, out of 196 URLs there are 177 internal links and 19 external links. So MD5Signature algorithm works well as compared to other two algorithms.

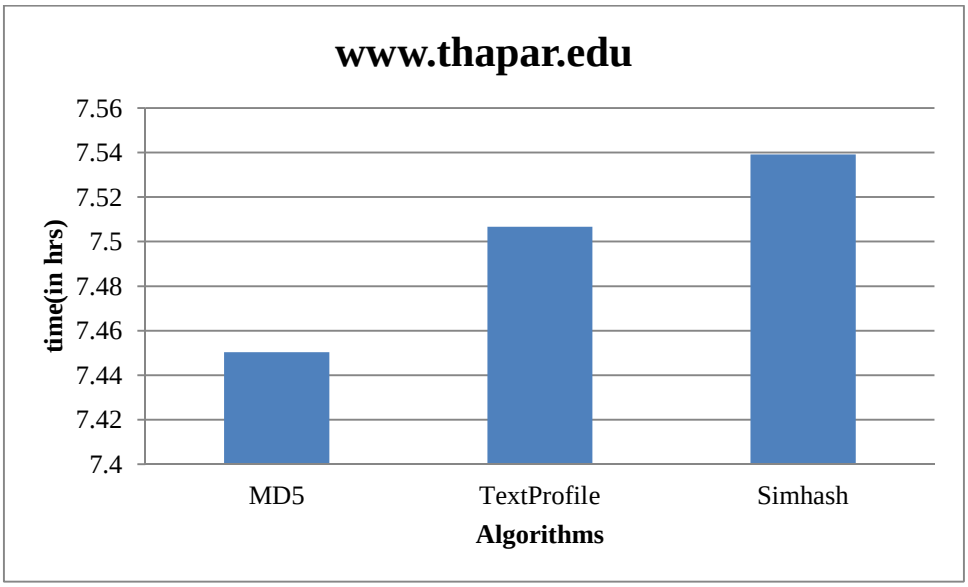

Fig 5.7: time (in hrs) taken by website (www.thapar.edu)

The time taken by the Nutch crawler to crawl website www.gr8ambitionz.com using MD5Signature as M, TextProfileSignature as T and Simhash algorithms as S as depicted in fig 5.8. As the site contains only 202 URLs, out of 202 URLs there are

180 internal links and 22 external links So Simhash algorithm works well as compared to other two algorithms.

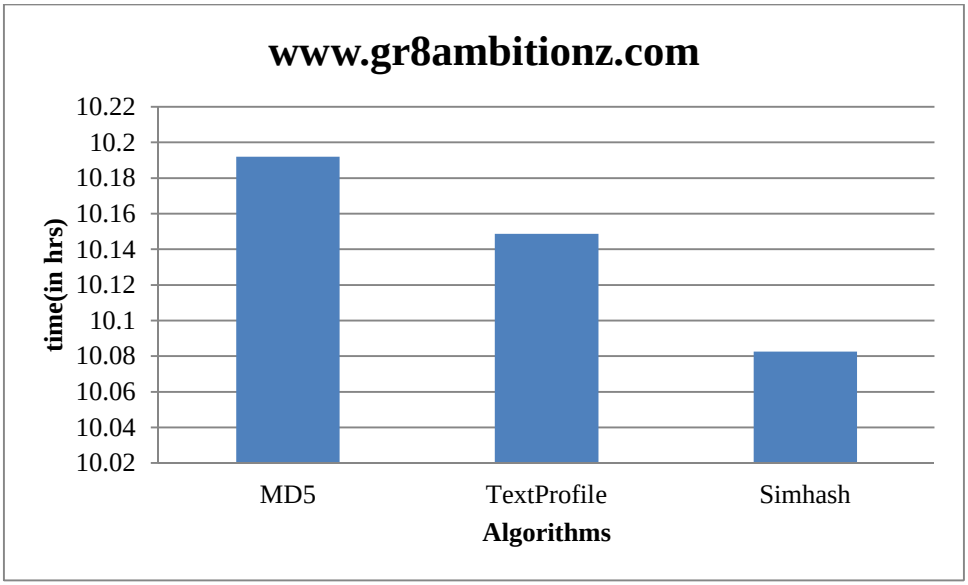

Fig 5.8: time (in hrs) taken by website (www.gr8ambitionz.com)

The time taken by the Nutch crawler to crawl website www.examrace.com using MD5Signature as M, TextProfileSignature as T and Simhash algorithms as S as depicted in fig 5.9. As the site contains only 1184 URLs, out of 1184 URLs there are 1178 internal links and 6 external links. So Simhash algorithm works well as compared to other two algorithms.

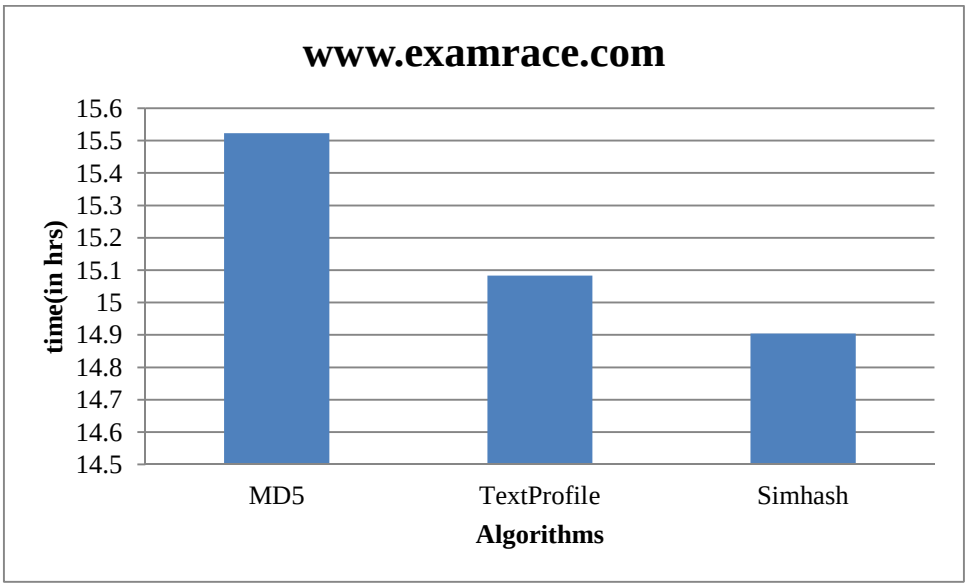

Fig 5.9 time (in hrs) taken by the website (amazon.com)

After comparing the time taken by the individual website to crawl using algorithms *i.e.* MD5 algorithms, TextProfileSignature algorithms and Simhash algorithm, we are comparing the time taken by the different websites to crawl using different algorithms as depicted in fig. 5.10.

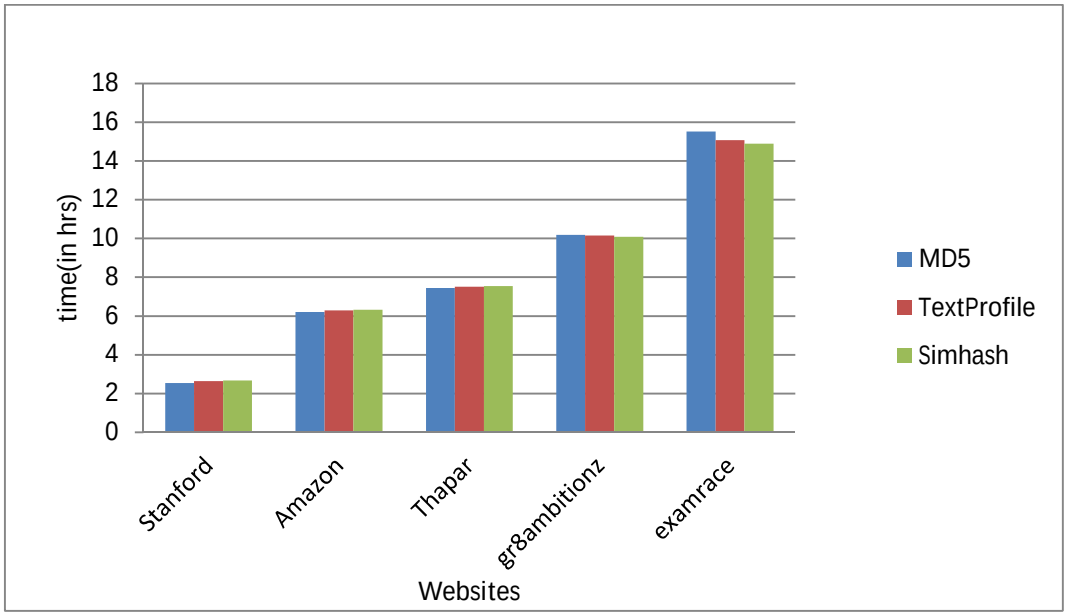

Fig 5.10: time (in hrs) taken by each website to crawling using different algorithms

As the number of URLs within the website increases, the time taken by the Simhash algorithm is less then as compared to the other two algorithms.

The number of duplicate URLs detected by the solr using the above mentioned algorithms is shown in table 5.4.

Table 5.4: No. of duplicate URLs detected by different algorithms

Algorithm	No. of duplicate URLs detected
MD5Signature	15
TextProfileSignature	49
Simhash	85

Chapter 6

Conclusion and Future Scope

6.1 Conclusion

From the literature survey and the experimental setup, it has been concluded that Simhash algorithm performs better as compared to MD5 Signature, and TextProfileSignature algorithms on the basis of time taken by the Nutch crawler to crawl the websites and for finding the count of duplicate entries.

The major contribution of this work is given below:

- The key contribution of this work is the experimentally proved comparative analysis between conventional approach (MD5 fingerprints and TextProfileSignature) and modernistic approach (locality sensitive hashing), taking one sample technique in each set. This is found that the locality sensitive hashing performs better in term of time taken to crawl and redundancy removal as compared to the conventional approach.
- The locality sensitive hashing also perform better in term of finding the number of duplicate URLs as compared to the conventional approach. The MD5 Signature algorithms find only those URLs which are exact duplicate of a particular URL.

6.2 Future Scope

This proposed work has focused on the comparative analysis of two different approaches but both the approaches are focused on syntactic similarity of web contents. However, many other recommendations hold ability of making the system more efficient or improving the quality of near-duplicate detection techniques. These are as follow:

- Finding duplicate detection techniques which focuses on semantic similarity of web contents instead of syntactic similarity [53] for eliminating duplicate content for web crawler.
- We can use machine learning techniques for near-duplicate document detection and add new feature vector analysis [26, 50, 52] to improve the

training set classification using which we can remove templates, menus and advertisements.

References

[1] S. Brin and L. Page, "The anatomy of a large-scale hypertextual Web search engine", *Computer Networks and ISDN Systems*, vol. 30, no. 1-7, pp. 107-117, 1998.

[2] “Desktop search engine market share” https://www.netmarketshare.com/ [as accessed on 20th Feb. 2016].

[3] Sun, Yang, Ziming Zhuang, and C. Lee Giles. "A large-scale study of robots. txt." *Proceedings of the 16th international conference on World Wide Web*. ACM, 2007.

[4] “Architecture of web crawler”, https://www.microsoft.com/en-us/research/wp-content/uploads/2009/09/EDS-WebCrawlerArchitecture.pdf. [as accessed on 25th June 2015].

[5] A. Broder, M. Najork and J. Wiener, "Efficient URL caching for world wide web crawling", *Proceedings of the twelfth international conference on World Wide Web - WWW '03*, 2003.

[6] R. C. Dharmik, T. V. Udapure and R. D. Kale, "Study of Web Crawler and its Different Types", *In proceeding to IOSR-JCE*, 2014.

[7] A. Arasu, J. Cho, H. Garcia-Molina, A. Paepcke and S. Raghavan, "Searching the Web", *ACM Trans. Inter. Tech.*, vol. 1, no. 1, pp. 2-43, 2001.

[8] R. Cai, J. Yang, W. Lai, Y. Wang and L. Zhang, "iRobot", *Proceeding of the 17th international conference on World Wide Web - WWW '08*, 2008.

[9] F. Menczer, G. Pant, P. Srinivasan and M. Ruiz, "Evaluating topic-driven web crawlers", *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval - SIGIR '01*, 2001

[10] P. Boldi, B. Codenotti, M. Santini and S. Vigna, "UbiCrawler: a scalable fully distributed Web crawler", Softw: Pract. Exper., vol. 34, no. 8, pp. 711-726, 2004.

[11] M. Najork and A. Heydon. High-performance web crawling. SRC Research Report 173, Compaq Systems Research Center, Palo Alto, CA, Sept. 2001.

[12] J. Tomlin, J. Edwards and K. McCurley, "An adaptive model for optimizing performance of an incremental web crawler.", In Proceedings of the 10th International World Wide Web Conference, pp. 106–113, 2001.

[13] H. Garcia-Molina and J. Cho, "Parallel crawlers", In Proceedings of the 11th International World Wide Web Conference, pp. 124-135 2002.

[14] A. Broder, S. Glassman, M. Manasse and G. Zweig, "Syntactic clustering of the Web", *Computer Networks and ISDN Systems*, vol. 29, no. 8-13, pp. 1157-1166, 1997.

[15] B. Alsulami, M. Abulkhair and F. Eassa, "Near duplicate document detection survey", *International Journal of Computer Science and Communications Networks*, vol. 2, no. 2, pp. 147--151, 2012.

[16] Z. Bar-Yossef, I. Keidar and U. Schonfeld, "Do not crawl in the DUST", *ACM Trans. Web*, vol. 3, no. 1, pp. 1-31, 2009.

[17] M. Henzinger, "Finding near-duplicate web pages", *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval - SIGIR '06*, 2006.

[18] W. Pugh, Monika H. Henzinger, “Detecting duplicates and near-duplicate files”, United States Patent, Patent No. US6658423B1, Dec 02, 2003.

[19] D. Meyerzon, S. Shoroff, F.S.Terek, S. Norin, “Method and system for detecting duplicate documents in web crawls”, United State Patent, Patent No. US6547829B1, Apr 15, 2003.

[20] F. Dend and D. Rafiei, "Approximate detecting duplicates for streaming data using bloom filters", *Proceedings of the 2006 ACM SIGMOD international conference on Management of data*, pp. 25--36, 2006.

[21] S. Brin, J. Davis and H. García-Molina, "Copy detection mechanisms for digital documents", *ACM SIGMOD Record*, vol. 24, no. 2, pp. 398-409, 1995.

[22] G. Di Lucca, M. Di Penta and A. Fasolino, "An approach to identify duplicated web pages", *Proceedings 26th Annual International Computer Software and Applications*, 1996.

[23] G. Manku, A. Jain and A. Das Sarma, "Detecting near-duplicates for web crawling", *Proceedings of the 16th international conference on World Wide Web - WWW '07*, 2007.

[24] A. Agarwal, H. Koppula, K. Leela, K. Chitrapura, S. Garg, P. GM, C. Haty, A. Roy and A. Sasturkar, "URL normalization for de-duplication of web pages", *Proceeding of the 18th ACM conference on Information and knowledge management - CIKM '09*, 2009.

[25] B. Alsulami, M. Abulkhair and F. Eassa, "Near duplicate document detection survey", *International Journal of Computer Science and Communications Networks*, vol. 2, no. 2, pp. 147--151, 2012.

[26] Z. Bar-Yossef, I. Keidar and U. Schonfeld, "Do not crawl in the DUST", *ACM Trans. Web*, vol. 3, no. 1, pp. 1-31, 2009.

[27] U. Manber, "Finding similar files in a large file system", *Usenix Winter*, vol. 94, pp. 1--10, 1994.

[28] M. S. Charikar, "Similarity estimation techniques from rounding algorithms", *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pp. 380--388, 2002.

[29] S. Brin, J. Davis and H. García-Molina, "Copy detection mechanisms for digital documents", *ACM SIGMOD Record*, vol. 24, no. 2, pp. 398-409, 1995.

[30] G. Manku, A. Jain and A. Das Sarma, "Detecting near-duplicates for web crawling", *Proceedings of the 16th international conference on World Wide Web*, pp. 141--150, 2007.

[31] N. Shivakumar and H. Garcia-Molina, "The SCAM Approach to Copy Detection in Digital Libraries", *D-Lib Magazine*, vol. 1, no. 5, 1995.

[32] M. Mathew, S. N. Das, T. R. Laxmi and K. Pramod, “A novel approach for near-duplicate detection of Web pages using TDW matrix”, *Proceedings of the*

9th international conference on Information Technology: New Generations (ITNG), pp. 121-126, 2012.

[33] C. Xiao, W. Wang, X. Lin, J. Xu Yu, “Efficient Similarity Joins for Near Duplicate Detection”, *Proceedings of the 17th international conference on World Wide Web, NY, USA,* pp. 131-140, 2008.

[34] Arun PR, Sumesh MS, “Near-Duplicate Web Page Detection by Enhanced TDW and simHash Technique”, *Proceedings of the international conference on Computing and Network Communications(CoCoNet)*, Trivandrum , pp. 765-770, 2015.

[35] Ernesto DI Iorio, Michelangelo Diligenti, Marco Gori, Marco Maggini, Augusto Pucci, “Detecting Near-replicas on the Web by Content and Hyperlink Analysis”, *Proceedings of the IEEE/WIC international conference on Web Intelligence*, pp. 249-255, 2003.

[36] A. Broder, S. Glassman, M. Manasse and G. Zweig, "Syntactic clustering of the Web", *Computer Networks and ISDN Systems*, vol. 29, no. 8-13, pp. 1157-1166, 1997.

[37] J. Cooper, A. Coden and E. Brown, "Detecting similar documents using salient terms", *Proceedings of the eleventh international conference on Information and knowledge management*, pp. 245--251, 2002.

[38] Michael O. Rabin, Fingerprinting by Random polynomial.

[39] A. Sasturkar, A. Roy, C. Haiti, S. Garg, A. Agarwal, K. Chitrapura, K. Leela and H. Copula, "URL normalization for de-duplication of web pages", *Proceedings of the 18th ACM conference on Information and knowledge management*, pp. 1987--1990, 2009.

[40] F. Chen, Y. Wu, H. Huang, X. Zhou and X. Zhang, "A Space-saving URL Duplication Removal Method for Web Crawler", *Journal of Information & Computational Science*, pp. 1195-1203, 2012.

[41] K. Bharat and A. Broder, "Mirror, mirror on the Web: a study of host pairs with replicated content", *Computer Networks*, vol. 31, no. 11-16, pp. 1579-1590, 1999.

[42] V. A. Narayana, P. Premchand, A. Govardhan, “A novel and efficient Approach for near duplicate page detection in Web crawling”, *Proceedings of the IEEE International Advance Computing Conference (IACC)*, pp. 1492-1496, Patiala, 2009.

[43] Jack G. Conrad, Xi S. Guo, Cindy P. Schriber, “Online Duplicate Document Detection: Signature Reliability in a Dynamic Retrieval Environment”, *Proceedings of the 12th international conference on Information and Knowledge management*, pp. 443-452, NY, USA, 2003.

[44] N. Joshi and J. Gadge, "Near Duplicate Web Page Detection using NDupDet Algorithm",*International Journal of Computer Applications*, vol. 61, no. 4, pp. 56-59, 2013.

[45] L. Pamulaparty, C. G. Rao and M. Rao, "XNDDF: Towards a Framework for Flexible Near-Duplicate Document Detection Using Supervised and Unsupervised Learning", *Procedia Computer Science*, vol. 48, pp. 228-235, 2015.

[46] D. Fetterly, M. Manasse and M. Najork, "On the evolution of clusters of near-duplicate web pages", *Journal of Web Engineering*, vol. 2, no. 4, pp. 228--246, 2003.

[47] R. Gupta, N. Duhan, A.K. Sharma and N. Aggarwal, “Query Based Duplicate Data Detection on WWW”, *International Journal on Computer Science and Engineering (IJCSE)*, vol. 02, no. 04, pp. 1395-1400, 2010.

[48] S. Zambom, H. Luiz, R. dos Santos Mello and M. Roisenberg, "Smart Crawler: Using Committee Machines for Web Pages Continuous Classification", *Proceedings of the 21st Brazilian Symposium on Multimedia and the Web*, pp. 125--132, 2015.

[49] “Nutch crawler” Available at: https://wiki.apache.org/nutch/NutchTutorial/ [as accessed on 06th Jan. 2016].

[50] “Apache Tomcat” Available at: http://tomcat.apache.org/tomcat-8.5-doc/index.html/ [as accessed on 12th Jan. 2016].

[51] “Apache Solr” Available at: http://lucene.apache.org/solr/resources.html#documentation/ [as accessed on 03th Feb. 2016].

[52] “Cygwin” Available at: http://cygwin.com/docs.html/ [as accessed on 06th Jan. 2016].

[53] Crawler Architecture, http://nlp.stanford.edu/IR-book/html/htmledition/crawler-architecture-1.html [as accessed on 10th Jan. 2016].

[54] Hephaestus, Article on web crawlers, Hephaestus Books, 2011.

[55] N. Heintze. “Scalable document fingerprinting”, *In Proceedings of the Second USENIX Electronic Commerce Workshop*, pages 191–200, Nov. 1996.

[56] L. Zambom Santana, R. dos Santos Mello and M. Roisenberg, "Smart Crawler", *Proceedings of the 21st Brazilian Symposium on Multimedia and the Web - WebMedia '15*, 2015.

[57] C. Olston, M. Najork, Web Crawling: Foundation and trends in information retrieval, Now Publishers Inc, 2010.

[58] Y.Bao, E.M. Bakker ,"Content Based Web Sampling", *JDCTA*, vol. 4, no. 1, pp. 43-68, 2010.

[59] M. Chau and H. Chen, "A machine learning approach to web page filtering using content and structure analysis", *Decision Support Systems*, vol. 44, no. 2, pp. 482-494, 2008.

[60] Y. Bernstein and J. Zobel, "Accurate discovery of co-derivative documents via duplicate text detection", *Information Systems*, vol. 31, no. 7, pp. 595-609, 2006.

List of Publications and Video URL

Communicated Paper:

[1] S. Panwar, V. Arora, "A survey of near-duplicate detection approaches for web crawler" in 5th International Conference on Advances in Computing, Communication and Informatics, Jaipur, 2016.

Link for thesis presentation

- https://www.youtube.com/channel/UCSZ7Rp9Bvu5Qqwc_ZsiRceQ?guided_help_flow=3