Web Mining and Content Analysis - Crawling and Indexing Web Content

SIDDHVED PHADKE, a1756264 - The University of Adelaide, Australia
SHUBHAM GUPTA, a1787223 - The University of Adelaide, Australia

Internet usage has increased a lot in recent times. Users can find their resources by using different hypertext links. This usage of Internet has led to the invention of web crawlers [28]. As the size of the web keeps on evolving, searching it for helpful data has become very difficult. The traditional crawling strategies cannot adapt to constantly developing web [29]. In this project we aim to study various crawling and indexing strategies.

Additional Key Words and Phrases: Web Crawling, Web Indexing, Search Engine

ACM Reference Format:
Siddhved Phadke and Shubham Gupta. 2020. Web Mining and Content Analysis - Crawling and Indexing Web Content. 1, 1 (October 2020), 17 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Web crawlers are small programs that browse the web on search engine's behalf. The crawlers are given starting set of URL's who's page they retrieve from the web typically for the purpose of web indexing. Early search engines used information retrieval algorithms and techniques but these were developed for relatively small and coherent collection such as newspaper articles. Web, on the other hand, is massive, less coherent, changes more rapidly and geographically spread [2]. Today, there are multiple algorithms and architectures which are used to crawl links from the web. For example authors in "An efficient approach for web indexing"[10] proposed an efficient method to crawl and index the links associated with the specified URL's. This paper first focuses on identifying the best method to crawl links and then builds efficient extraction to identify metadata. The authors in "Information retrieval from the web and application of migrating crawler" [29] proposed a novel concept of Migrating crawlers (parallel crawlers). It is a mobile process which works on behalf of its user to interact with web servers to gather information. The initialization of crawling in web is difficult. Different web crawlers such as Semantic Web crawler, Smart web crawlers has been reviewed based on information fetched by various crawlers. While reviewing different types of crawlers, Smart Web Crawlers gives reduced crawling time as mentioned. The author Viv Cothey [8], investigated the reliability of collecting data by web crawling. He found that web crawling by search engines is intentionally biased and selective.

These are one of the few topics which will be covered in this report. We aim to study and evaluate different types of web crawling architectures and web crawling algorithms which have been introduced. We will point out the the advantages of one system over other and gaps in the system.

Authors' addresses: Siddhved Phadke, a1756264 - The University of Adelaide, West Avenue, Northfield , Adelaide, Australia, a1756264@student.adelaide.edu.au; Shubham Gupta, a1787223 - The University of Adelaide, Eliza Street, Gilberton , Adelaide, Australia, shubham.gupta@student.adelaide.edu.au.

2 TYPES OF WEB CRAWLERS

Depending on how the web pages are crawled and how successive web pages are retrieved to reach the next pages, various types of web crawlers are possible. Below we have provided some types used in state of the art papers:

2.1 Breadth First Crawlers

Breadth first crawlers start with a limited collection of pages and then navigate to other pages by following links provided in a breadth-first like fashion. This is usually how one can build a major search engine or a large repository [27]. In real life, web pages the web pages use variety of policies instead of just strictly following breadth first. For example, the most important pages will be crawled first by the crawler.

2.2 Incremental Web Crawlers

Updating an existing list downloaded pages, instead of restarting the crawl from scratch is basically what a Incremental Web Crawler does [23]. A methodology is used to determine if the page has been modified since the last it was crawled. A crawler, based on some set of crawling cycles, that will continually crawl the entire web. A model is used which is adaptive in its nature, which evaluates which pages should be reviewed for changes. Because of this model, high freshness is achieved and significantly low peak load performance. The basic overview of a incremental web crawler is shown in figure 1.

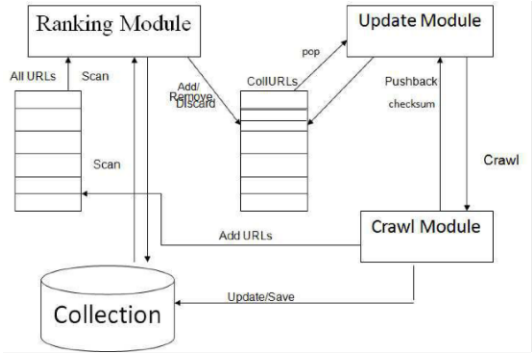

Fig. 1. Incremental Web Crawler [28]

2.3 Form Focused Web Crawler

Form Based Crawler tackles the sparse distribution on the Network of forms. Form Crawler[26] By restricting the search to a specific subject, form crawler avoids crawling through ineffective paths. Some other ways by which crawler avoids ineffective paths are- learning the features of the links and paths which lead to pages containing any sort of search-able forms, And using a suitable stop criteria. The crawler uses two classifiers to direct its search: the page and the link classifiers. Later, one other classifier is used to remove useless types with the form classifier. The page classifier is trained to classify pages as taxonomic subjects. The strategy used by Form Focused Web Crawler is the same as the one used by Best First crawler.

2.4 Focused Crawler

Hypertext Classifier, which was developed by authors in [4, 5] was the basis for the creation of Focused Crawler. It is mainly composed of three main components -

- To make relevant decisions on pages to be crawled and expanded, a classifier is used.
- To measure the centrality of pages crawled to evaluate visit priorities, a distiller is used.
- At last, a crawler with priority controls dynamically re-configurable by the above mentioned classifier and distiller.

The goal of the focused crawler is to provide a simplified alternative to overcome the problem of low-ranking immediate pages relevant to the subject at hand. To conduct an exhaustive search recursively (upto a certain depth) is the main idea behind it. The exhaustive search start from the relatives of a top priority or highly ranked page.

2.5 Hidden Web Crawler

A collection of web based data, which cannot be accessed necessarily through hyperlinks but only trough web-based search forms. Still large amount of data on the web remains in the database and can only be accessed by posting suitable requests or by filling out the aforementioned search forms. Work on Hidden Web Crawler has been happening for more than a decade. The reason work on Hidden Web Crawler started was to find a way to access information various online databases that were normally hidden behind search forms. The crawlers used generally today, can only crawl "PIW" or Web which is only Publicly Index-able. It is basically a collection of web pages that can only be accessed by following hyperlinks present in search pages and also in forms that require user permission or corresponding registration. Because of this, huge amount high quality data which maybe be behind the search forms or registrations can be overlooked or missed.

2.6 Parallel Crawlers [6]

The Parallel web crawler produces different types of operations and can work parallelly with other crawlers. As the web is growing enormously it is not feasible to rely on only one crawler. Either crawler can be managed centrally or can be dispersed over the network as well. Study suggests that crawlers over distributed network are far more superior than multithreaded crawlers and stand-alone crawlers. This superiority is based on the overall scalability, efficiency, and throughput of the crawlers. When computed over dispersed network architecture and reduced network load, parallel web crawlers provide best results out there compared to other web crawlers. Parallel web crawler mainly focused on the Random-Access Memory of the computer or machines when there is no disk access. Hence it utilises main memory of the system to run the crawling algorithm so that it can be accessed over the single machine or network of machines where the crawler crawls through the web pages which are being saved in the cache memory of the system.

2.7 Distributed Web Crawler

Distributed Web Crawling is a distributed computing technique in which internet search engines use a number of computers (or systems) to index the internet through web crawling. This crawler runs on network of workstations. Such a structure allows the users decide their own computing and bandwidth resources needed for traversing through the web. Because of the rising and significantly complex nature of the web, indexing the web is very challenging. So therefore, it becomes necessary to run the crawling process in parallel to finish in reasonable amount of time. By distributing the strain of these activities over several machines, massive cost which might be expended on maintaining

the computing clusters can be avoided. Individual URLs (Uniform Resource Locator) are distributed to several web crawlers that retrieves the web pages in conjunction with each other. The central indexer receives the downloaded pages from the crawler, where the links are extracted and sent to the crawler again via the URL server. As a result, the hardware requirements for crawling process is significantly decreased and also receives a significant improvement in the overall download speed and the reliability in the process[14].

2.8 Distributed Focused Web Crawler

Mining data from a web server and from a web database are two different things. This is because mining from database is designed to the gather the data which is unique from a particular website as explained by the authors in detail in [1]. Also if anyone tries to scrape or collect large amount of data in short amount of time, that can be detected as a cyber thread and can be banned or stopped from linking to the web server. The authors of the paper[1] also mentioned how it can be avoided by proposing a novel crawling method, which at the same time is cheaper and more efficient than the web crawlers used traditionally. It is mentioned that the using any 4^{th} generation languages like Java, Python etc, these kind of web crawlers can be developed as these languages support multi thread. Something that enables the system to run many processes at once is called Multi Threading. This web crawler strategy proposed by the authors results in the following :

- Mining speed is significantly improved.
- A lot safer than using single thread of web crawler.

3 WEB CRAWLING ARCHITECTURES

In this section, we aim to understand and explain some of the Web Crawling Architectures used in state of the art papers. We have picked five most popular architectures which have used over the years.

3.1 Web Crawling Architecture from "MultiCrawler: A Pipelined Architecture for Crawling and Indexing Semantic Web Data" [11]

The authors listed a number of requirement which according to them needed to be taken into account when one is designing a web crawler. Which are:

- **Performance and scalability** -
  - To keep up the constant growth taking place in the data source.
  - Also to manage data which is present on the web scale.
  - We should focus on performance-oriented architecture rather than something else.
  - By incorporating new hardware, without a fundamental overhaul, the device should be able to scale up.
- **Utilizing data from different formats and disparate sources**-
  - The system needs to index data from multiple web sources
  - At the same time the system has to syntactically transform the indexed data and arrive at an integrated dataset.

The authors here have used a pipe-lined architecture which lead to a considerable performance improvements. They have also defined a software pipeline which is used for Semantic Web data crawling and indexing. Executing the different steps in parallel was the main idea behind such a pipeline which lead to performance improvements[11]. We can see the architecture in Figure 2. The process of crawling was logically split into five phases, which are namely **Fetch, Detect, Transform, Index, and Extract** as seen in Figure 2.

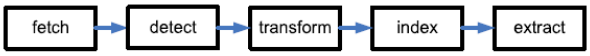

Fig. 2. Architecture of Web Crawler from [11]

3.2 Web Crawling architecture from "Topical Web Crawlers: Evaluating Adaptive Algorithms" [20]

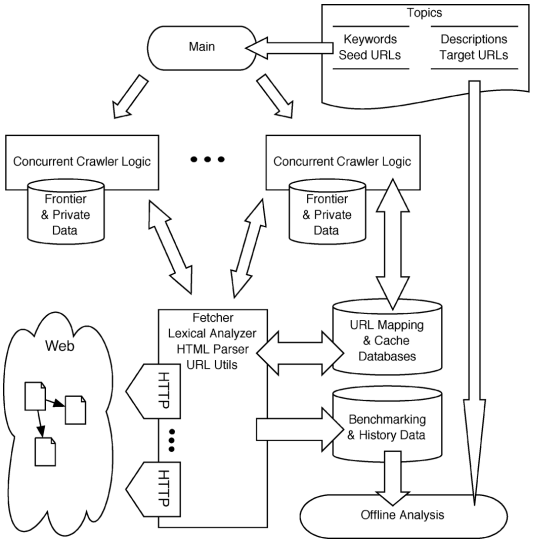

Fig. 3. Architecture of Web Crawler from [20]

In order to encapsulate any specific crawling algorithm, the following system architecture was designed. With this the any crawler can be directly plugged into the system through a general interface. The architecture of the proposed crawler is illustrated by the authors in Figure 3. To optimize efficiency and at the same time not affecting evaluation, all the crawling module are made to share utilities and public data structure[20].

3.3 Web Crawling Architecture from "An Efficient Approach for Web Indexing of Big Data through Hyperlinks in Web Crawling"[10]

The system proposed by authors in this paper uses an algorithmic process for Web Crawling. Using depth first manner to crawl a web is the most effective way. This allowed the crawled links by the crawler to be acquired in a continuous hyperlink manner. Google's idea[10] was one of the backdrop for the development of the proposed system. We can see the entire proposed crawling architecture in the Figure 4. The key features of the proposed system are as follows [10]:

- Traversing through the web, it needs to be identified and indexed.
- New pages should be be accessed from the old pages with the help of hyperlinks provided.
- For future use metatag content in the database should be stored
- Time-consuming calculations such as Page-Rank should be always avoided.
- To control the contents, the robots needs to be altered so that the system is able to access it.

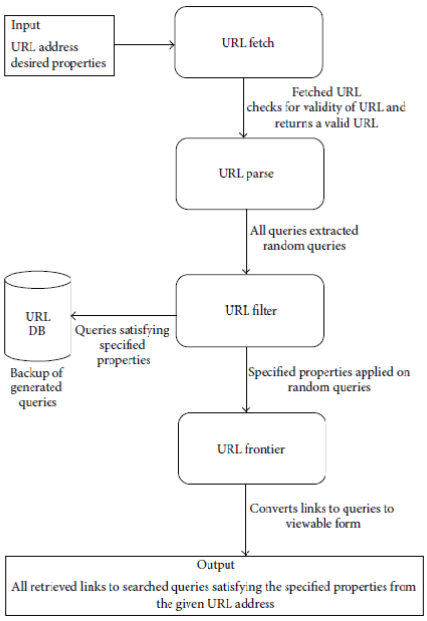

Fig. 4. Architecture of Web Crawler from [10]

The block diagram(4) consists of the mechanism by which the documents defined by the analyzer are crawled from a web URL. The method suggested in this paper by the authors guarantees that it is within the web crawling ethics guidelines. The framework proposed in this paper by the authors, maintains track of the documents which are crawled by the crawler and at the same time meta-tag extraction is updated. Because of this overlap is prevented and also no duplication of the records. Which hyperlinks to crawl from the given links, the crawler lets the analyzer decide that. It is followed by verification of URL specified, done with URL application programming interface validator. Then hyperlinks are accessed associated with the respective URL. Then the crawler either terminated the process or selects a different path based on condition of the properties which are specified earlier. For further purposes of research, the generated queries are entirely backed up. Along with their metadata extracted, the links are stored in the database. One of the main advantages of this architecture is its information integration with the metatag extraction which is simultaneously occurring.

3.4 Web Crawling Architecture from "Web Crawler: Extracting the Web Data"[29]

The most significant component of a web search engine such as Google or Bing is a a web crawler. Over the years with which the size of the web is growing, same growth has been seen in the development of a web crawler. A list of URLs is available when using a web crawler. Each one of the available URL is called a "seed"(See Figure 5). The crawler visits each one of the listed URLs. Now, all the hyperlinks on the website are detected by the web crawler and are added to the list of the URLs to visit subsequently. Crawl frontier is the term used for this list. There are policies and rules in place for the URLs that needs to be visited. The parser and the generator downloads different pages from the internet and are subsequently stored in the search engines database. The search engine can now access all the URLs one by one

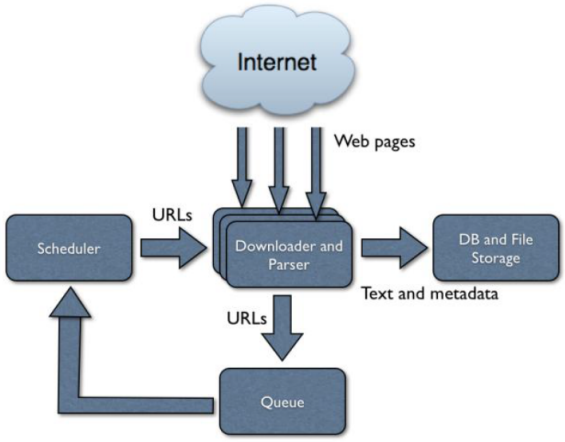

Fig. 5. Web Crawler Architecture from [29]

which were placed on queue and later scheduled by the scheduler. Now according to the requirement, all the related file and links being searched can be made available. Web crawlers find the related links for the search engines using relevant algorithms and use them further.

3.5 Web Crawling Architecture from "A Semantic Focused Web Crawler Based on a Knowledge Representation Schema"[12]

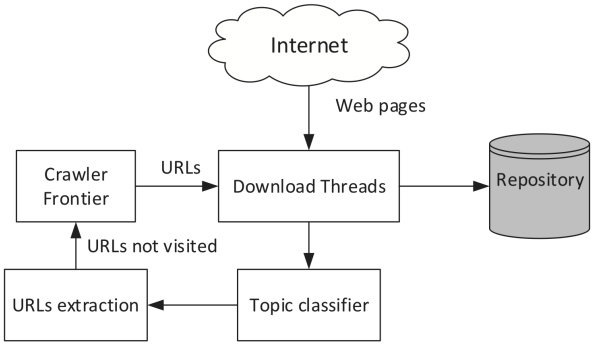

Fig. 6. General Focused Web Crawler Architecture from [12]

The architecture of a web crawler used by the authors is composed of three main components:

- **URL Frontier** - Which basically stores all the URLs which needs to be crawled by the crawler.
- **Page Downloader** - From frontier URLs, the page downloader retrieves all the web pages and subsequently parses them.
- **repository** - All the downloaded web pages are stored in repository.

Some of the vast amount of resources on the network may be unrelated to the area of interest. One of the reason for the preference of the focused web crawlers to retrieve web pages is this. Machine Learning Classification algorithms is one of the technique on which a focused web crawler is based on which is used to find relevant web pages and at the same time adding it to the local database [12]. A topic classifier module is added on top of a traditional crawler to form a focused web crawler(Figure 6). To identify related web pages, this module is featured-based, modelling an input target domain. If the web page is categorised positively, its URLs in the frontier module are extracted and queued. The classification module in some focused web crawlers is based on metrics evaluated by document similarity which help us separate non-related and related web pages of a given domain. But the expressiveness of the content in the web pages are not taken into account in such approaches. Which in simple words mean that their semantic content is not explored[12].

4 WEB CRAWLING ALGORITHMS

In this section we aim to describe and evaluate five different crawling algorithms:

- Breadth-First
- Best-First
- PageRank
- Shark-Search
- InfoSpiders

Our selection is intended to be a broad representation of the crawler algorithms reported in the state of the art papers.

4.1 Breadth-First

A Breadth-First crawler is the easiest technique for Crawling. This algorithm was explored as early as 1994 in the WebCrawler [25], in Efficient crawling [7] and in Breadth-First [22]. It utilises the frontier as a queue for FIFO, crawling links in the the order they are located in. Note that the crawler can add only one link from a crawled page when the frontier is full. The Breadth-First crawler is illustrated in 7 and the algorithm is shown in 8. Breadth-First is used as a baseline crawler to give a lower bound to more sophisticated algorithms. It is used as a baseline because no awareness of the subject is used in it.

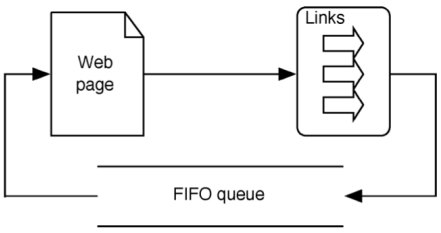

Fig. 7. A Breadth First Crawler [20]

```
Breadth-First (starting_urls) {
   foreach link (starting_urls) {
      enqueue(frontier, link);
   }
   while (visited < MAX_PAGES) {
      link := dequeue_link(frontier);
      doc := fetch(link);
      enqueue(frontier, extract_links(doc));
      if (#frontier > MAX_BUFFER) {
         dequeue_last_links(frontier);
      }
   }
}
```

Fig. 8. Psuedocode of the Breadth First Crawler [20]

4.2 Best-First

Best-First crawlers have been studied in Efficient crawling [7] and The Shark-Search Algorithm [13]. The basic idea is that the best link is selected for crawling given a boundary of links according to some estimation criteria. On the basis of increasingly sophisticated link estimation parameters, various Best-First strategies of increasing complexity and (potentially) efficacy could be developed. Figure 9 illustrates a Best-First crawler, while Figure 10 offers a simplified pseudocode of the Best-First-Search (BFS) algorithm. The sim() function returns the cosine similarity between topic and page.

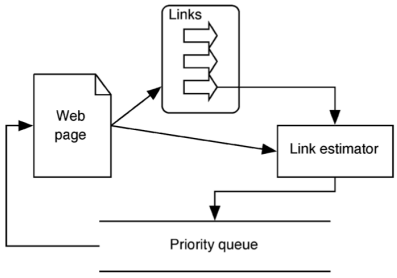

Fig. 9. A Best-First Crawler [20]

4.3 PageRank

PageRank was proposed in The anatomy of a large-scale hypertextual web search engine by [3] as a possible model of user surfing behavior. A page's PageRank reflects the possibility of a random surfer (one who follows links randomly from page to page) being on that page at any given time. The score of a page is recursively based on the scores of the pages that point to it. Source pages distribute their PageRank across all of their outlinks. PageRank was meant to be used in tandem, as originally suggested by [3], with the ranking of recovered sets of documents with content-based criteria. This is in fact how PageRank is used in the Google search engine. We can see from Figure 11, PageRank requires a recursive calculation, so its computation can be a very resource-intensive process before convergence. One should

```
BFS (topic, starting_urls) {
   foreach link (starting_urls) {
      enqueue(frontier, link, 1);
   }
   while (visited < MAX_PAGES) {
      link := dequeue_top_link(frontier);
      doc := fetch(link);
      score :=  sim(topic, doc);
      enqueue(frontier, extract_links(doc), score);
      if (#frontier > MAX_BUFFER) {
         dequeue_bottom_links(frontier);
      }
   }
}
```

Fig. 10. Best-First Algorithms [20]

```
PageRank (topic, starting_urls, frequency) {
   foreach link (starting_urls) {
      enqueue(frontier, link);
   }
   while (visited < MAX_PAGES) {
      if (multiplies(visited, frequency)) {
         recompute_scores_PR;
      }
      link := dequeue_top_link(frontier);
      doc := fetch(link);
      score_sim :=  sim(topic, doc);
      enqueue(buffered_pages, doc, score_sim);
      if (#buffered_pages >= MAX_BUFFER) {
         dequeue_bottom_links(buffered_pages);
      }
      merge(frontier, extract_links(doc), score_PR);
      if (#frontier > MAX_BUFFER) {
         dequeue_bottom_links(frontier);
      }
   }
}
```

Fig. 11. Pseudocode of PageRank crawler [20]

recalculate PageRanks any time a URL needs to be accessed from the frontier in the ideal situation. Instead, to improve efficiency and computational overheads it is recomputed only at regular intervals. With a different link assessment function, the PageRank crawler can be seen as a variant of Best-First. The algorithm is illustrated in Figure 11. Note that in order to compute PageRank, the PageRank algorithm must preserve the data structure of visited pages in addition to the frontier. This buffer can contain at most MAX BUFFER pages.

4.4 Shark-Search

Shark-Search [13] is a more aggressive version of Fish-Search [9]. The crawlers search more extensively in Fish-Search in areas of the Web where relevant pages have been found. At the same time, the algorithm stops looking in regions which do not generate appropriate pages. Shark-Search offers mainly two improvements over Fish-Search. In contrast to the binary relevance feature in Fish-Search, it uses a continuously valued function to calculate relevance. Furthermore, Shark-Search has a more refined definition of possible scores for links at the frontier of the crawl. Anchor text, text surrounding the links, and inherited score from ancestors (pages pointing to the page containing the links) affect the

possible score of links. With a more sophisticated link assessment feature, the Shark-Search crawler can be seen as a variant of Best-First. Figure 12 shows the pseudocode of the Shark-Search algorithm. The parameters d and r represent respectively the maximum depth and the relative importance of inherited versus neighbourhood scores.

```
Shark (topic, starting_urls) {
   foreach link (starting_urls) {
      set_depth(link, d);
      enqueue(frontier, link);
   }
   while (visited < MAX_PAGES) {
      link := dequeue_top_link(frontier);
      doc := fetch(link);
      doc_score :=  sim(topic, doc);
      if (depth(link) > 0) {
         foreach outlink (extract_links(doc)) {
            score = (1-r) * neighborhood_score(outlink)
                  +  r    * inherited_score(outlink);
            if (doc_score > 0) {
               set_depth(outlink, d);
            } else {
               set_depth(outlink, depth(link) - 1);
            }
            enqueue(frontier, outlink, score);
         }
         if (#frontier > MAX_BUFFER) {
            dequeue_bottom_link(frontier);
         }
      }
   }
}
```

Fig. 8. Pseudocode of the Shark-Search crawler.

Fig. 12. Pseudocode of the Shark-Search crawler [20]

4.5 InfoSpiders

In InfoSpiders [16–19], An adaptive agent population uses evolving query vectors and neural nets to scan for pages related to the subject to determine which links to pursue. The idea is illustrated in Figure 13. This evolutionary approach

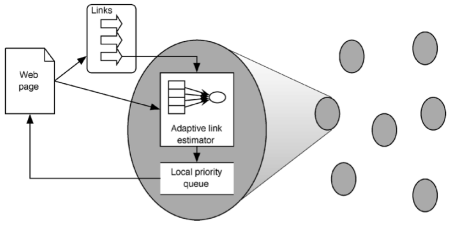

Fig. 13. A population of InfoSpiders [20]

utilises a fitness measure dependent on similarity as a criterion for local selection. The original algorithm as seen in [18] was simplified and implemented as a crawler module in [21]. Due to its extreme local behaviour and the lack of any memory, this crawler was outperformed by BFS. Once a link was followed, the agent could only access pages connected from the new tab. Since then, a number of improvements have been made to the original algorithm while retaining its ability to learn link estimates via neural networks and focus its search through selective reproduction towards more

promising areas. The resulting novel algorithm is schematically described in the pseudocode of Figure 14. The agents of

```
IS (topic, starting_urls) {
   agent.topic := topic;
   agent.energy := 0;
   agent.fsize := MAX_BUFFER;
   agent.frontier := starting_urls;
   insert(agent, population);
   while (visited < MAX_PAGES) {
      foreach parallel agent (population) {
         link := pick_and_dequeue(agent.frontier);
         doc := fetch(link);
         newenergy = sim(agent.topic, doc);
         agent.energy += newenergy;
         learn_to_predict(agent.nnet, newenergy);
         foreach outlink (extract_links(doc)) {
            score = ALPHA * newenergy +
                    (1 - ALPHA) * agent.nnet(doc, outlink);
            enqueue(agent.frontier, outlink, score);
         }
         if (#agent.frontier > agent.fsize) {
             dequeue_bottom_links(agent.frontier);
         }
         delta := newenergy - sim(topic, agent.doc);
         if (boltzmann(delta)) {
            agent.doc := doc;
         }
         if (agent.energy > THETA and pop_size < MAX_POP_SIZE) {
            child.topic := expand(agent.topic, agent.doc);
            (child.energy, agent.energy) := (0, 0);
            (child.fsize, agent.fsize) := half(agent.fsize);
            (child.frontier, agent.frontier) := split(agent.frontier);
            child.nnet := extend(agent.nnet, child.topic)
            insert(child, population);
         }
      }
   }
}
```

Fig. 14. Pseudocode of InfoSpiders algorithm, each agent executes as concurrent process in actual implementations. [20]

InfoSpiders are autonomous and crawl in parallel. Each agent 's adaptive representation consists of a list of keywords (initialised with keywords from the subject) and a neural net used to evaluate new ties. Each neural net input unit receives a count of the frequency at which the keyword takes place in the vicinity of each link to be crossed, weighted to give greater significance to keywords that occur near the link (and maximum in the anchor text). There is a single output unit. One of the simplifications we have maintained in this implementation is that there are no hidden layers in the neural networks, so they consist of simple perceptrons. What this means is that while the output unit of a neural net computes a nonlinear sigmoidal function of the input frequencies, only linear separations of the input space can be learned by the neural network. For each link considered as an input, the neural net output is used as a numerical quality estimate. By predicting the similarity of the new page, the neural net can be trained to boost the link estimates, provided the inputs from the page that contained the link that led to it. InfoSpiders have the unique ability to adjust the link-following behavior during a crawl by associating relevant estimates with complex patterns of keyword frequencies across links through such a basic reinforcement learning technique. While InfoSpiders was the first crawling algorithm to use reinforcement learning [16], others authors have considerably expanded this in papers [15, 15, 24].

5 CHALLENGES

5.1 No Headers Found for URL Based Web Crawler

- In this study, we have found that most of the web servers are not willing to go hand in hand with the web browsers. For example, we have checked a "dummy" browser with Opera and Internet explorer which is able to send the content on the web page which are requested. The results found were very exhausting as the requested web page displays everything except status line and headers.
- It seems that the pages that has been downloaded on the web browser with headers and status line but not able to display on the web servers.
- Any real websites, presumably due to misconfigured codes or misconfigured firewalls, display the same misbehaviour as our dummy web servers. To obtain right content, the Web crawler should be prepared to display status lines and headers.

5.2 Accessing Wrong Dates in headers for URL Based Web Crawler

- There are several numbers of computers which do not have updated their date and time which applies for time zones as well. Lot of people around the world use these tricks as well to prolong their paid software's trail duration.
- In the recent study, we have found that over 17% of web servers around the world has returned with the no last-modification details, and dates which are not updated since the creation of the web only.
- This error impacts a lot on the Last-Modified field in the web server's response, which in this case can not be used for further calculation of the bandwidth and time.
- However, if a web server clocks the wrong time but just for few minutes or for few hours, there might be a possibility that the web server clock has misrepresented which leads to the wrong calculation over the web server time stamping which will eventually goes displaying the wrong dates and status line in the headers.

5.3 Detection of Unwanted blogs, forums

- The world of web contains number of online blogs, online forums which have large number of small repositories as well of the data, which when called can be displayed by the web server.
- There are number of pages which consists of the individual user's messages in the repositories which are useful for the content searched by the user for particular topic.
- The main purpose of these web pages is to display the related set of topics and work as typical examples which are messages from the technical team of the servers which explains solutions to the problems.
- These typical examples can be specifically for the device or the hardware specifications of the systems. However, these typical examples are not useful as they are inefficient and contains very short and brief information of the content.
- The main reason these examples are inefficient as they are not as precised and less functional than expected.
- Also, these typical examples are not available as detailed summary of web content, but the complete follow-up conversation might be sometimes useful.
- To avoid these un-wanted blogs and forums, a good scheduling algorithm is useful which will try to download the related and best pages first related to content being searched.

5.4 Content duplication detection

- The ubiquity of finding the same content on the web is very high. Study shows over 30% of the duplicity of same web content[30].
- Hashing algorithm are used to avoid storing of the web pages twice with identical content.
- Hashing algorithm provide the reliable and scalable source of information which is being tested every time someone uploads new content on the web.
- In this algorithm, it checks line by line words of the content and shows duplication error while uploading data for similar content besides if uploader has input of different images and format.
- Although, the content can not be stored on the web, it can always be downloaded from non-credible sources.
- The duplicated content downloading may lead to waste of network resources which may further lead the web-crawler disoriented and mislaid for downloading the correct web pages.

5.5 Web content available with slow and erroneous pages

- Web pages can be distributed into two parts which are Dynamic and static pages. This can be followed by the main difference where dynamic pages are those web pages which can accept user input in terms of storing those user inputs on web server's cache memory.
- On the other hand, static web pages do not follow this rule as user is not allowed to give input on the web page. As static web pages are not allowed to apply any changes by the user, they are not supposed to use cache memory of the web page server. This makes static web pages slower as compared to dynamic web pages.
- This error is generated due to citing different number if pages and source of information or in some cases it can be error of programming.
- Due to these erroneous and slow pages, it causes timeout in web crawler to download the latest web pages.

5.6 General web crawler challenges

- With the help of recent study there are several problems which were identified for various crawling architecture.
- Several web crawlers are based on the crawling and downloading the pages which are being visited by the user. Due to continuous growth of the web, downloading crates bottleneck feature on the downloading side.
- Duplication of the content has also raised due to less integrity in the web content uploaded over the web which creates unnecessary load over the network.
- Recent study suggests, due to bottleneck the downloading rate of the crawler has been reduced.
- Also due to reduced throughput of the overall functionality of the web crawling, crawler has to revisit the already downloaded pages to check for updating the recent content that has been uploaded.
- Hence, in order to keep updated database for search engine, crawlers must constantly update the already downloaded web pages.
- The Efficiency of the web crawlers can be assessed with minimizing the traffic over network and still able to crawl and download relatable content. The more efficient way for web crawling is to index the web pages which has been crawled before so that it only revisit the web page if it gets updated content with help of indexes provided to web pages.
- Increasing the size of web indexes does not improve quality of serach pages if it has to crwal twice to get the updated content of the same page.

6 FUTURE WORK

Here, we list some work which can be pursued in future to potentially improve the web crawling architectures.

- **Multi-crawler pipe-lined architecture** -
  - Bottleneck has been the main issue which has not been resolved yet. This web crawler is sincerely based on generating results according to Resource Description framework (RDF) rather than focusing on the HTML and XML web data.
  - Future work for this architecture can be done in the field of reducing the amount of HTML and XML data sources and increasing the amount of RDF sources which works as linkage between the source of the web content to use URI allowing the structured and semi-structured data to be shared across different applications.
  - In this web crawler architecture it can be achieved in the transform phase where it is easy to process less pages which results in the increased throughput of the overall architecture.
  - Also, this crawlers architecture consists of future work where it is tested on the large number of datasets to perform continuous crawling for longer time.
- **Topical web crawlers**-
  - Due to disappointing adaptation of the algorithms there is a good scope for the future work.
  - Firstly, in topical web crawlers Infospider algorithm has provided great experimental results by showing its re-usability, great scalability, and shorter time to crawl for small as well as large data sets.
  - But it is parallelly becoming worse when it comes to large frontier sizes which may lead to reproducing the same downloaded pages for several time reducing the efficiency of the crawler architecture.
  - Due to obvious limitations in hardware, bandwidth and time and regularly strained length of crawls expected results has not been generated by this architecture.
  - So future work, for this architecture consists of improving the quality of the hardware, dedicated bandwidth of network and able to crawl for longer period of time this web crawler architecture may lead to perform well in future.
- **Semantic Focused Web Crawler**
  - As the system started working and in progress the number of crawled pages is also increasing in the “An Efficient Approach for web indexing of Big data through Hyperlinks in web crawling” architecture defined
  - Increased crawl pages show that performance is bound to the maximum efficiency of the crawler.
  - The proposed architecture mainly intended to build database pages and links with help of World Wide Web.
  - It also focuses on updating the web pages to keep the current content on the page which has been crawled before.
  - Future work for this architecture can be done in bandwidth reduction to synthesize the system so that the system is approachable for the next level of the links.
  - In semantic web crawler, it focuses on the Knowledge representation Schema (KRS) which possess qualities like it can model any domain, it is less complex.
  - In fact, Semantic Focused Web Crawler uses KRS as a shield to construct a semantic analysis used as user input.
  - For Future work, The input web page corpus for the KRS construction can be discovered with newer approach which will be helpful to extensive evaluation of the topics of relevant topic document.

7 CONCLUSION

The rate at which web data is generated now, we need a web crawling system which can handle this much data. In this report our aim was to study and evaluate the different types of architectures and systems, As web crawlers are one or if not the most important aspect of all search engines. As seen above, if is not difficult to build a web crawler but to build a web crawler which is efficient and optimized and a intelligent web crawler is still something that can be achieved. We found some gaps in the research and proposed some future work which could help build better web crawling and indexing systems. The reader of this report will have clear understanding of what a web crawling and indexing system does, the advantages of using a particular system over other and what work can be done in the future.

ACKNOWLEDGMENTS

To Dr. Wei Zhang, for her encouragement and expert advise throughout this project. Prof. Javen Shi, Dr Ehsan Abbasnejad, Guanhua Wang and Joshua Groot for their continuous online support.

REFERENCES

[1] Harry T Yani Achsan and Wahyu Catur Wibowo. 2014. A fast distributed focused-web crawling. *Procedia Engineering* 69, 1 (2014), 492–499.
[2] Arvind Arasu, Junghoo Cho, Hector Garcia-Molina, Andreas Paepcke, and Sriram Raghavan. 2001. Searching the web. *ACM Transactions on Internet Technology (TOIT)* 1, 1 (2001), 2–43.
[3] Sergey Brin and Lawrence Page. 1998. The anatomy of a large-scale hypertextual web search engine. (1998).
[4] Soumen Chakrabarti. 2002. *Mining the Web: Discovering knowledge from hypertext data.* Elsevier.
[5] Soumen Chakrabarti, Martin Van den Berg, and Byron Dom. 1999. Focused crawling: a new approach to topic-specific Web resource discovery. *Computer networks* 31, 11-16 (1999), 1623–1640.
[6] Junghoo Cho and Hector Garcia-Molina. 2002. Parallel crawlers. In *Proceedings of the 11th international conference on World Wide Web*. 124–135.
[7] Junghoo Cho, Hector Garcia-Molina, and Lawrence Page. 1998. Efficient crawling through URL ordering. (1998).
[8] Viv Cothey. 2004. Web-crawling reliability. *Journal of the American Society for Information Science and Technology* 55, 14 (2004), 1228–1238.
[9] Paul ME De Bra and RDJ Post. 1994. Information retrieval in the World-Wide Web: Making client-based searching feasible. *Computer Networks and ISDN Systems* 27, 2 (1994), 183–192.
[10] R Suganya Devi, D Manjula, and RK Siddharth. 2015. An efficient approach for web indexing of big data through hyperlinks in web crawling. *The Scientific World Journal* 2015 (2015).
[11] Andreas Harth, Jürgen Umbrich, and Stefan Decker. 2006. Multicrawler: A pipelined architecture for crawling and indexing semantic web data. In *International Semantic Web Conference*. Springer, 258–271.
[12] Julio Hernandez, Heidy M Marin-Castro, et al. 2020. A Semantic Focused Web Crawler Based on a Knowledge Representation Schema. *Applied Sciences* 10, 11 (2020), 3837.
[13] Michael Hersovici, Michal Jacovi, Yoelle S Maarek, Dan Pelleg, Menachem Shtalhaim, and Sigalit Ur. 1998. The shark-search algorithm. An application: tailored Web site mapping. *Computer Networks and ISDN Systems* 30, 1 (1998), 317–326.
[14] Dhiraj Khurana and Satish Kumar. 2012. Web crawler: A review. *International Journal of Computer Science & Management Studies* 12, 1 (2012), 401–405.
[15] Andrew McCallum, Kamal Nigam, Jason Rennie, and Kristie Seymore. 1999. A machine learning approach to building domain-specific search engines. In *IJCAI*, Vol. 99. Citeseer, 662–667.
[16] Filippo Menczer. 1997. ARACHNID: Adaptive retrieval agents choosing heuristic neighborhoods for information discovery. In *MACHINE LEARNING-INTERNATIONAL WORKSHOP THEN CONFERENCE-*. MORGAN KAUFMANN PUBLISHERS, INC., 227–235.
[17] Filippo Menczer and Richard K Belew. 1998. Adaptive information agents in distributed textual environments. In *Proceedings of the second international conference on Autonomous agents*. 157–164.
[18] Filippo Menczer and Richard K Belew. 2000. Adaptive retrieval agents: Internalizing local context and scaling up to the Web. *Machine Learning* 39, 2-3 (2000), 203–242.
[19] Filippo Menczer and Alvaro E Monge. 1999. Scalable web search by adaptive online agents: An infospiders case study. In *Intelligent Information Agents*. Springer, 323–347.
[20] Filippo Menczer, Gautam Pant, and Padmini Srinivasan. 2004. Topical web crawlers: Evaluating adaptive algorithms. *ACM Transactions on Internet Technology (TOIT)* 4, 4 (2004), 378–419.
[21] Filippo Menczer, Gautam Pant, Padmini Srinivasan, and Miguel E Ruiz. 2001. Evaluating topic-driven web crawlers. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*. 241–249.

[22] Marc Najork and Janet L Wiener. 2001. Breadth-first crawling yields high-quality pages. In *Proceedings of the 10th international conference on World Wide Web*. 114–118.

[23] András Nemeslaki and Károly Pocsarovszky. 2011. Web crawler research methodology. (2011).

[24] Tadhg O'Meara and Ahmed Patel. 2001. A topic-specific Web robot model based on restless bandits. *IEEE Internet Computing* 5, 2 (2001), 27–35.

[25] Brian Pinkerton. 1994. Finding what people want: Experiences with the WebCrawler. In *Proc. 2nd WWW Conf., 1994.*

[26] Sandeep Sharma and Ravinder Guide Kumar. 2008. *Web-Crawling Approaches in Search Engines.* Ph.D. Dissertation.

[27] Vladislav Shkapenyuk and Torsten Suel. 2002. Design and implementation of a high-performance distributed web crawler. In *Proceedings 18th International Conference on Data Engineering*. IEEE, 357–368.

[28] Mini Singh Ahuja Dr Jatinder Singh and Bal Varnica. 2014. Web Crawler: Extracting the Web Data. *International Journal of Computer Trends and Technology* 13, 3 (2014), 132–137.

[29] Niraj Singhal, RP Agarwal, Ashutosh Dixit, and AK Sharma. 2011. Information retrieval from the web and application of migrating crawler. In *2011 International Conference on Computational Intelligence and Communication Networks*. IEEE, 476–480.

[30] Xiao Xu, Weizhe Zhang, Hongli Zhang, and Binxing Fang. 2010. Efficient Distributed Web Crawling Utilizing Internet Resources. *IEICE TRANSACTIONS on Information and Systems* 93, 10 (2010), 2747–2762.