Web Pages Retrieval by Using Proposed Focused Crawler

Dunia Hamid Hameed[1] and Soukaena Hassan Hashem[2]
Department of Computer Science, College of Sciences, University of Technology, Baghdad-Iraq.
[1]E-mail: dunia_h_h@yahoo.com.
[2]E-mail: soukaena.hassan@yahoo.com.

Abstract

"Focused Crawler" is designed to visit a part of the web to collect documents that are related to only a particular topic. The objective of focused crawler is to identify good links that lead to target required documents, and to avoid branches that don't lead to the required topic. There is a number of motivations for designing focused crawler such as: fetching relevant data from the web with simplified data indexing, personalizing the human-computer interaction, making the system adaptive with each user, needing for a tool to change the searching strategy, keeping the freshness of the web pages and filtering the links to keep track focusing on the user's preference. In this paper, we will explain two methods to retrieve web pages by using traditional crawler and proposed focused crawler. We make several experiments and it shows that proposed focused crawler is more efficient than traditional crawler in retrieving the desired web pages.

Keywords: Crawler, Web retrieval, Focused crawler, Traditional crawler.

Introduction

"General web crawler" (also called Universal Crawlers) starts with one or more URLs of the first page, gets the first page's URL list; while crawling the web page, the web crawler extracts new URLs from the current page, and puts them into the queue of URLs waiting to be fetched until it meets the system's stop condition. A General crawler follows every links it finds, resulting in the download of many irrelevant and duplicated pages [1] [2].

General web crawler sometimes is unsuitable to crawl the web efficiently (like crawling through web forums) for numerous reasons as follows [3]:

1. It loses useful pages because the crawling done without understanding the relevance in meaning of the crawled pages.
2. It may crawl redundant pages or useless pages.

Focused crawler aims to collect data which fit specific requirement, therefore the developing of such crawler is important issue. The reason is always there is a need for retrieving relevant pages with high quality that can be managed, maintained, updated and retrieved easily by using efficient mechanisms [4].

In other words, Focused web crawler is used for retrieving subset of web to gather web pages that are relevant to a specific topic and the irrelevant web page will be skipped to reach the links of desired pages. Thus focused crawling can be used to generate data for an individual user. Focused crawling is a necessary concept for IR to collect documents of one domain [5] [6] [7].

By comparing with general crawling, focused crawling permits access to special data and precludes some problems like computational and financial bottlenecks resulted from full web crawling [8].

A focused crawler characterized by using a ranking function to determine which outgoing link will be traversed next with considering that it only crawls relevant pages. Focused crawler is maintaining reasonable dimensions of the index. The concentration focused crawler in reducing the crawl boundary is attractive because "a recognition that covers a single galaxy can be more realistic and useful than trying to cover the entire universe" [9] [10].

Crawler systems used as tools for focused crawling, for shopping systems implementation and for supporting services like added-value services on the Internet (portals, personalized and mobile services, etc.) [11].

The work on focused crawling assumes two properties [12] [13]:

1. Linkage Locality: Pages that are related to same domain will be linked together. This property used by the crawlers of early search engines which crawl the pages deeply and stop when they find irrelevant page. This strategy has disadvantage when the closely related pages are not linked directly.
2. Sibling Locality: This property means that the outlinks of the page are mostly point to similar pages of the same domain. This page called Hub and the pages that pointed by Hub called authorities.

A focused crawling algorithm starts with loading a page and extracting the links, by estimating relevance of the pages according to links and based on keywords. The links will be crawled one after the other and increase the scope of its work with investigation of the content of the page. Three issues are important for focused crawling [6]:

1. Relevance estimation of crawled web page.
2. Determination of potential URLs that might be relevant pages.
3. Specifying the next link to crawl by ranking and ordering relevant pages.

There are three focusing techniques that could be used in preserving the focusing of the crawler [14]:

1. Link structure analysis: is depend on the hypothesis that two similar pages may be pointed by the same page, or they are links in the content of same document which means there is similarity between them.
2. Content analysis: is looking at the word similarity between documents. This is based on the hypothesis that two documents related to the same topic will use the same words.
3. Hybrid approaches which mixes the two types. This lead to the concept of a distance metric.

One of techniques of Focused crawlers is to use old crawled pages to discover the relevance of new pages. It depends on efficient modeling to understand the context. Another approach is the integration of genetic algorithm with another artificial intelligence technique which is the ant algorithms to enhance focused crawler performance. This achieved by the optimization of genetic operators like selection, mutation and crossover [15] [16].

The efficiency of focused crawling can be measured by finding the ratio between the relevant web pages and the total number of web pages that retrieved by the focused crawler. If the result is high, it means that the crawler is efficient and it reached to desired web pages that are more than irrelevant web pages during the crawling process [17].

Most approaches that manipulate Focused Crawling have the following disadvantages [18]:

- Crawler skips the non-target pages because it is specialized to specific domain. These pages might be limited in increasing the ability of generalization.
- If the seeds of crawling are not closely related, the focused crawler will not work properly.
- Focused the capability to reach the target pages.

Evaluation Metrics

To evaluate the performance of IR system, there are two metrics: precision and recall. They are considered as most frequent and basic measures for evaluating the effectiveness of IR systems [19].

Precision (P) is the fraction of retrieved documents that are relevant as the following equation [19]:

Precision= Relevant Items Retrieved/Retrieved Items. .... (1)

Recall (R) is the fraction of relevant documents that are retrieved as the following equation [19]:

Recall = Relevant Items Retrieved/Relevant Items. .... (2)

Related Works

Li et al. [4] suggested a proposal called “an Enhanced-Form Focused Crawler (E-FFC)” for a particular domain of Web Databases (WDBs). This crawler dealt with restrictions to perform efficient retrieval with good coverage rate. Experiments of the E-FFC conducted on several pages of different domains and certified that it performed better than other focused crawlers of specific domain of deep web form in terms of the crawling robustness, rates of harvesting and coverage.

Zunino et al. [20] suggested a strategy to design focused crawler used for multiple usages such as searching, monitoring, and Open Source Intelligence (OSINT). The objective of this crawler was to be suitable with the requirements of intelligent analysts. It had specific flexibility according to the required aspects and crawling method. Moreover, it can learn and adapt with the analyst’s need. The method implementation was depending on combined environment semantic networks, ontologies and text mining. Experiments demonstrated that adaptive mechanism was effective.

Gouriten et al. [21] presented adaptive systems for focused crawling. It suggested focused crawling method applicable to different situations. It proposed an algorithm, which allowed the system to be able to identify relevant subsystems optimization. Although the information was available, it showed that there was difficulty in exploration. The experiments demonstrated efficient performance of greedy algorithm and the importance of adaptive estimation of the crawler frontier.

Gossen et al. [22] suggested combining the web and social media by using focused crawler that dealt with web retrieval of a topic such that the pages are relevant and updated. It used the stream of fresh social media content for controlling the crawler. The results showed that crawler made use of stream data of social media contents which are continuous and fresh to adapt its behavior automatically to get the most promising information source.

Implementation of the Proposed Crawlers

Description of Robot.txt File

Robots.txt is a text file exists in the root of the hierarchy of any web page. It is created by the owners of the web pages to determine restrictions to prevent the web crawlers from accessing their web page either partly or entirely. These restrictions are like instructions with special format. Each crawler has to adhere with this file before retrieving anything. If this file is missing, web crawlers consider that the web page’s owner permits crawling the page entirely. Algorithm (1) produces checking for a given URL to check if the crawler allowed to crawl it or not. Ethical crawlers retrieve the robot file robot: txt and specify which parts are disallowed for crawling. These parts prohibited either for reasons of privacy or the owner thinks that these parts are irrelevant to the crawler’s need. The comments can be written after character (#).

There are two terms used in robot: txt file: User Agent which is a term describing the programs that allowed to visit the web page and Disallow statements which are used to specify the paths that are not allowed to visit. Examples of Robot.txt file are as the following:

Example (1): This example informs all crawlers that they have right to retrieve all files because the wildcard (*) specify all crawlers:

```
User-agent: *
Disallow:
```

The same result can be obtained with an empty or missing robots: txt file.

Example (2): This example informs that all crawlers are prevented from downloading the web page:

```
User-agent: *
Disallow: /
```

Example (3): This example to show how it is possible to list multiple crawlers with their specific restrictions:

```
User-agent: X
Disallow: /dir1/
User-agent: Y
Disallow: /
User-agent: *
Disallow: /dir2/
```

Algorithm (1) describes how the crawler can check the Robot.txt file for a specific (URL).

Algorithm (1) Check Robot File
INPUT: URL to check if it is allowed to crawl or not *URL*
OUTPUT: Boolean value Flag.
STEPS:

```
1- Begin
2- Extract the host name of URL and save it in
   variable Host
3- Open connection for reading of robot file of
   URL
4- Read robot file and create list of disallowed
   links Disallow Links
5- Check for comments and delete them if
   exist
6- Remove leading and trailing spaces from
   disallow paths
7- Add disallow paths to list Disallow List
8- Check Disallow Links to know if the
   crawling is allowed for the specified URL
   URL
      if URL in Disallowed Links then
            Flag=FALSE
      else
            Flag=TRUE
      end if
9- Return Flag
10- End
```

Design of Traditional Crawler

This section describes the design of Traditional Crawler which is an important part in the General search engine (also called horizontal search engine). In the crawling process of the Links, there are two strategies in retrieving the links from the web page:

- Depth-first crawling: retrieving the first link on the first page then extracting the links and follows the first link of the new page and so on until reaching the end of all the links and traversing them.
- Breadth-first crawling: retrieving each link on the page before continuing with the links of the first page then follows all the links on the first page and so on until traversing all the links.

Traditional Crawler uses Breadth-first crawling manner and adheres with Robot protocol to know which links of web site are prohibited from crawling.

As described in algorithm (2), the crawler starts by initializing the list of URLs that will be crawled. Then by retrieving URL from the list, it begins the crawling process which includes: fetching the web page, parsing its contents, extracting the URLs from the web page, adding the links to the list of URL that will be crawled and it continues until the stop condition becomes true. The stopping condition depends on exhaustion all the links. Fig.(1) shows the steps of Traditional Crawler in Flowchart.

Algorithm (2) Traditional Crawler algorithm
INPUT: Seed URL.
OUTPUT: Text file to save URLs (URLs:txt).
STEPS:

```
1- Begin
2-Create List of URLs ToCrawlList and
  initialize it with link of Seed URL
3- Create List URLs to save the results
4- Perform Crawling
  Flag1 = Check Robot File (Seed URL) (call
  algorithm (1))
  if (Flag1 = True) then
  while (To Crawled List not empty) do
  - Get URL from To Crawl List and save it
    in variable Current URL
  - Flag2 = Check Robot (Current URL) (call
    algorithm (1))
    if (Flag2=True) then
  - Parse the downloaded web page retrieving
    all the links and adding them To Crawled
    List
  - Extract links of Current URL page
  - Add extracted links to list To Crawl List
  - Add Current URL to the list URLs
      end if
      end while
      else
      Print ("This URL not allowed to be
      crawled") and return emplty list
      end if
5-Print List of URLs (URLs) to URLs: txt file
6- End
```

The Proposed Focused Crawler

This section explains the design of the Proposed Focused Crawler which can be used in Vertical search engines. The proposed crawler depends on set of keywords that represent a description for the desired domain to crawl in it. Algorithm (3) describes the general steps of Proposed Focused Crawler. It includes calling of three algorithms (Check URL, Check Robot, Focused Crawling Process) as shown later. At the end of crawling process, the crawler will get a list of URLs that meet the required domain. Fig.(2) describes the flowchart of the proposed Focused Crawler.

Algorithm (3) Focused Crawler algorithm
INPUT: Seed URL *Seed URL*, list of keywords kewords and Maximum URL *Max URL*.
OUTPUT: File called (URLs.txt) contains the results of crawling the web.
STEPS:

```
1- Begin
2-Create a file for saving crawling results URLs:txt
3- Create list to save the matched links MatchedList
4- Flag 1 = Check URL (SeedURL) (call algorithhm (4))
5- Flag 2 = Check Robot File (SeedURL) (call algorithm (1))
6- Perform Focused crawling process
    if (Flag 1 = TRUE) and (Flag 2 = TRUE) then
    Matched List=Crawling Process (Seed URL, keywords, Max URL) (call algorithm (5))
    Save Matched List to the file URLs.txt
    else
    Print (Invalid URL, Enter another URL)
    end if
7- End
```

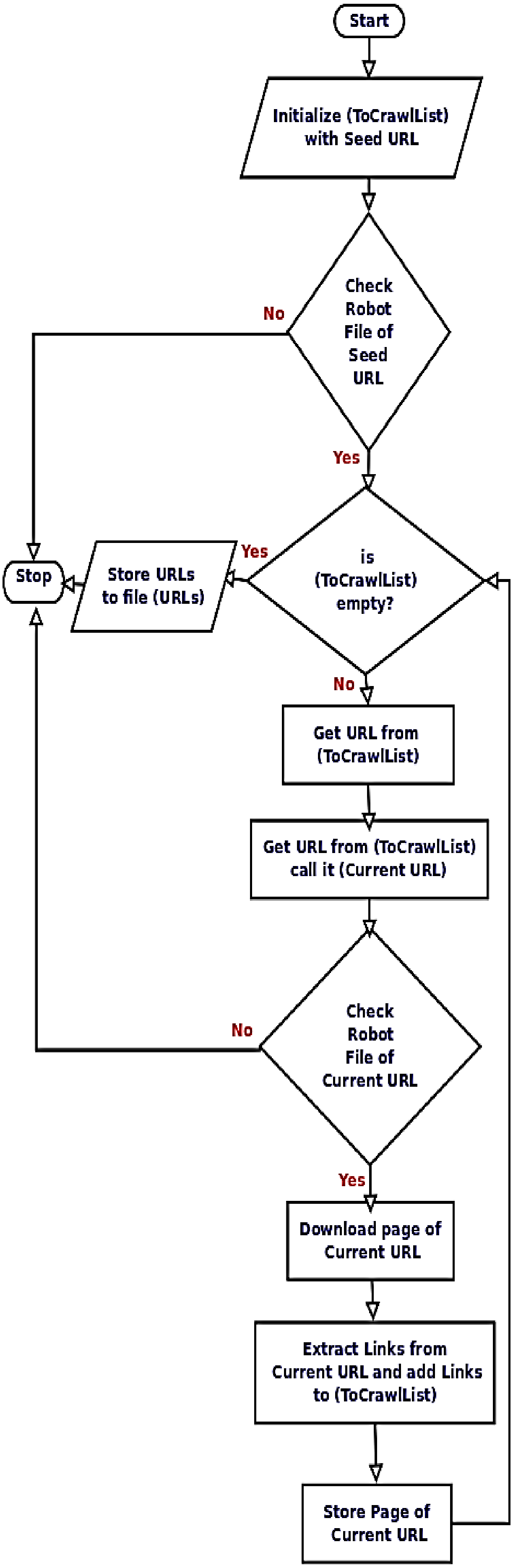
Start
Initialize (ToCrawlList) with Seed URL
Check Robot File of Seed URL
No
Yes
is (ToCrawlList) empty?
Yes
Store URLs to file (URLs)
Stop
No
Get URL from (ToCrawlList)
Get URL from (ToCrawlList) call it (Current URL)
Check Robot File of Current URL
No
Yes
Download page of Current URL
Extract Links from Current URL and add Links to (ToCrawlList)
Store Page of Current URL

Fig.(1): Flowchart of Traditional Crawler.

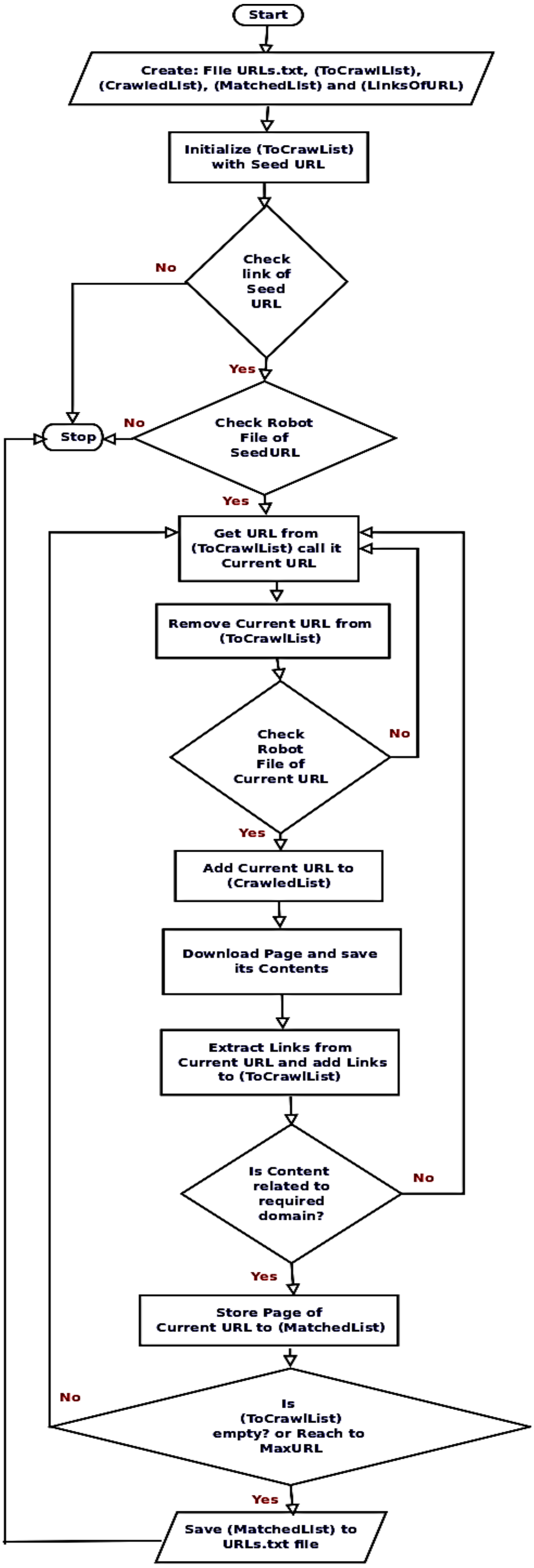
Start
Create: File URLs.txt, (ToCrawlList), (CrawledList), (MatchedList) and (LinksOfURL)
Initialize (ToCrawList) with Seed URL
Check link of Seed URL
No
Yes
Stop
Check Robot File of SeedURL
No
Yes
Get URL from (ToCrawlList) call it Current URL
Remove Current URL from (ToCrawlList)
Check Robot File of Current URL
No
Yes
Add Current URL to (CrawledList)
Download Page and save its Contents
Extract Links from Current URL and add Links to (ToCrawlList)
Is Content related to required domain?
No
Yes
Store Page of Current URL to (MatchedList)
Is (ToCrawList) empty? or Reach to MaxURL
No
Yes
Save (MatchedList) to URLs.txt file

Fig.(2): Flowchart of Proposed Focused Crawler.

At the beginning, Proposed Focused Crawler needs to check the validation of URL of seed document as in algorithm (4) and also it needs to check the Robot file to check if URL is allowed to crawl or not as in algorithm (1).

Algorithm (4) Check URL
INPUT: Seed URL URL.
OUTPUT: Boolean variable called (Flag).
STEPS:

```
1- Begin
2-Convert link URL to lowercase
3- Check the format of URL
    if (URL format is valid) then
    Flag=TRUE
    end if
    Return Flag
4- End
```

After checking the seed URL, Proposed Focused Crawler uses algorithm (5) that makes crawling process. The input to this algorithm is (Seed URL), List of keywords and the required number of URLs. During the Crawling process, four lists need to be created:

1. List to save the links that will be crawled called *(To Crawl List*).
2. List to save the links that crawled by the crawler called (*Crawled List).*
3. List to save the links that match the required domain which is called (*Matched List*). This list of URLs can be indexed later by using Indexing method of the vertical search engine. By this strategy the search engine can ensure that the content of its index is related to specific domain and it is not crawling all the links immediately.
4. List to save the links that extracted from the contents of crawled page called (*Links Of URL*).

Crawling Process starts by initializing the list (*To Crawl List*) with the seed document. Then iterate through the links of *To Crawl List* until it becomes empty or the crawler retrieves the required URLs.

Algorithm (5) Focused Crawling process
INPUT: Seed URL Seed URL, list of keywords kewords and Maximum URL Max URL.
OUTPUT: File called (URLs: txt) contains the results of crawling the web.
STEPS:

```
1-Begin
2- Create lists (To Crawl List, Crawled List,
   Matched List and Links Of URL)
3- Crawling Process
  - Add URL Seed URL to the list To Crawl
    List while ((size of To Crawl List > 0) or
    (size of Crawled List !=Max URL)) do
  - Get a link from To Crawl List and save it in
    variable URL
  - Remove the link URL from To Crawl List
    Flag = Check Robot File URL
         if (Flag = TRUE) then
  - Add the link to crawled list Crawled List
  - Download the page of URL and save the
    result in Contents
  - Links Of URL = Links Extraction (Contents,
    Crawled List) (call algorithm (6))
  - Add Links Of URL to the list To Crawl List
  - Check Keywords with the Contents
    if (Contents are related to keywords) then
                  Add link URL to match list
         Matched List
         end if
         end if
         end while
         Return Matched List
4- End
```

In each iteration, the crawler performs the following operations:

1. Get a Link from the list To Crawl List, call it Current URL and remove this link from the list To Crawl List.
2. Check the *robot.txt* file of the Current URL, if it is allowed to be crawled then it will continue in crawling. If it is prohibited from crawling, the crawler will get another URL.
3. Add the Current URL to the list *Crawled List.*
4. Open a Connection to the page of *Current URL* and save its contents.
5. Extract the links of the page contents.
6. Check the contents of the page, if it is related to desired domain then the link will added to *Matched List.*

Crawling Process algorithm uses two algorithms:

1. Check Robot File (Algorithm (1)) to verify that the crawler is not in *Disallowed list* of the page.
2. Link Extraction (Algorithm (6)) to extract links from web page contents.

To extract the links from the content of the crawled page, the Proposed Focused Crawler uses Link Extraction algorithm (6). This algorithm takes the contents of the page and the list of crawled URLs as input and produces a list of links (*Links Of URL*). At the beginning, the algorithm creates a pattern to check the links with this pattern. The algorithm performs several steps as the following:

Pattern Creation: the pattern is a general description of a regular expression that can be used to find a matching of sequence of characters in the page contents with the specified pattern. The pattern contains set of normal characters, wild characters and quantifiers. The algorithm uses (<a\\s+href\\s*=\\s*\"?(.*?)[\"|>]) as a pattern. The pattern can be explained as following:

<a	Search about string "<="
\\s+	Search about one space character or more
href	Search about the string "href" which represents the link of the page
\\s*	Search about zero space character or more
=	Search about the character "="
\\s*	Search about zero space character or more
\"?	Search about zero quote character or one
(.*?)	Search about zero of any character or more until other part of the pattern occurred and save the result in a group
[\"|>]	Search about quote character or greater than (">") character

- Filtering the Empty links to protect the crawler from wasting time.
- Skipping the links that contain only (#) character. Page anchors that permit links to be made to a specific part of a page.
- Skipping mailing links which are used for specifying an e-mail link in a web page.
- Ignoring Java Script links, which are adding interactivity to the page.
- Convert the form of the link to fully qualified form. There are three forms of the links:

  1. Fully Qualified Form as the following URL: (https://www.java.com/en/about/).
  2. Absolute Form: the URL omitting the "host" portion with the slash (/) exist at the start of the form. The slash denotes that the URL is "absolute". Absolute URL is as: (/en/about/).
  3. Relative Form: URL omitting the "host" portion of the URL and without leading slash. So the URL is considered to be a "relative". Relative, in the realm of URLs, means that the address of URL is relative to the URL on which the link is found in it, as: (en/about/).
- Remove the sequence (www) from the link.
- Ignore the links that exist in Crawled List to prevent crawler from re-downloading.

If the links passed all the steps of testing, the link is valid and it will be added to the list *Links Of URL*. The algorithm will stop after extracting all the links. This algorithm used during crawling process and it increases the quality of crawling.

Algorithm (6) Links Extraction
INPUT: text of the URL Contents and Crawled List.
OUTPUT: List contains the links of the page Links Of URL.
STEPS:

```
1- Begin
2- Create a pattern of the URL
3- Check the contents of the page with the
   pattern of URL
4- Create a list for the extracted links Links Of
   URL
5- Check the text to find links.
   while (not end of Contents) do
   while (valid pattern found Link) do
     - Ignore the empty links
     - Ignore the links which are page anchors
       (#)
     - Ignore the links which is for mail
     - Ignore the Java Script links
     - Convert Absolute and Relative links to
       fully qualified links
     - Remove (www) from link
     - Ignore links that are crawled
       CrawledList
       end while
     - Add the valid link to list of links Links
       Of URL
       end while
6- Return list of links LinksOfURL
7-End
```

Experimental Results

This section is dedicated to study the performance of the proposed crawlers. To validate the effectiveness of this method, standard recall and precision metrics will be used for comparing proposed Focused crawler with Traditional crawler. The Experiments conducted to test the proposed methods; they are six experiments with the same query. The queries are about the sorting algorithms.

Tables ((1) and (2)) show the results of evaluation of Traditional and Proposed Focused respectively. Several notes can be concluded from the following tables:

- The Traditional Crawler gives recall higher than Proposed Focused Crawler because it retrieves more web pages and it didn’t deal with the relevancy to the required domain.
- The Proposed Focused Crawler gives precision higher than Traditional Crawler because it retrieves more relevant pages. It ignores the pages that are unrelated to the given domain.
- The recall value for each query is higher than precision value in the Traditional crawler results.
- The precision value of each query is higher than recall value in the Focused crawler results.

Table (1)
Traditional Crawler Evaluation.

No.	***Query***	***Precision***	***Recall***
1	time, bubble, sort	70	82
2	sort, quick	82	85
3	quick, heap, space	77	84
4	Shell, sort, time	79	86
5	heap, space, sort	76	84
6	data, sort, time, shell	71	86

Table (2)
Proposed Focused Crawler Evaluation.

No.	***Query***	***Precision***	***Recall***
1	time, bubble, sort	80	81
2	sort, quick	88	82
3	quick, heap, space	85	81
4	shell, sort, time	86	84
5	heap, space, sort	85	83
6	data, sort, time, shell	88	82

Conclusions

From this research, several remarks were noticed and concluded. The most important ones are: Proposed Focused Crawler increases the performance of crawling, since it produces related web pages to the information need, this method makes visiting the links not random process but depending on the relevancy between domain and page contents and Focused crawling is useful in keeping the retrieved web pages fresh and updated.

References

[1] Q. Bai, G. Xiong, Y. Zhao, and L. He, “Analysis and detection of bogus behavior in web crawler measurement”. Procedia Computer Science, 31(0), 1084 – 1091, 2014. URL:http://www.sciencedirect.com/science/article/pii/S1877050914005407.

[2] Y. Zhou, Q. Zhang, X. Huang, and L.Wu, “A patternbased selective recrawling approach for object-level vertical search”. ACM. In Proceedings of the 22nd ACM international conference on Conference on information &#38; knowledge management, CIKM ’13, 1441–1450, New York, NY, USA, 2013. URL http://doi.acm.org/10.1145/2505515.2505707.

[3] A. Sachan, W. Lim, and V. Thing. “A generalized links and text properties based forum crawler”. In Proceedings of the The 2012 IEEE/WIC/ACM International Joint Conferences on Web Intelligence and Intelligent Agent Technology–01, WI-IAT’ 12, 113–120, Washington, DC, USA, 2012. IEEE Computer Society.URL http://dl.acm.org/citation.cfm?id=2457524.2457671.

[4] Y. Li, Y.Wang, and J. Du. “E-ffc: an enhanced formfocused crawler for domain-specific deep web databases”. Journal of Intelligent Information Systems, 40(1), 159 –184, 2013. URL http://dx.doi.org/10.1007/s10844-012-0221-8.

[5] C. Groc. “Babouk: Focused web crawling for corpus compilation and automatic terminology extraction”. Web Intelligence, 497–498. IEEE Computer Society, 2011. URL http://dblp.uni-trier.de/db/conf/webi/webi2011.html#Groc11.

[6] J. Rawat. “A study of focused web crawlers for semantic web”. (IJCSIT) International Journal of Computer Science and Information Technologies, 4(4), 398–402, 2013.

[7] H. Liu, E. Milios, and J. Janssen. “Probabilistic models for focused web crawling”. In Proceedings of the 6th Annual ACM International Workshop on Web Information and Data Management, WIDM’ 04, 16–22, New York, NY, USA, 2004. ACM. URL http://doi.acm.org/10.1145/1031453.1031458.

[8] A. Pirkola and T. Talvensaari. “Addressing the limited scope problem of focused crawling using a result merging approach”. ACM. In Proceedings of the 2010 ACM Symposium on Applied Computing, SAC’ 10, 1735–1740, New York, NY, USA, 2010. URL http://doi.acm.org/10.1145/1774088.1774459.

[9] J. Chen, R. Power, L. Subramanian, and J. Ledlie. “Design and implementation of contextual information portals”. ACM. In Proceedings of the 20th International Conference Companion on World Wide Web, WWW ’11, 453–462, New York, NY, USA, 2011. URL http://doi.acm.org/10. 1145/1963192.1963359.

[10] F. Abkenari and A. Selamat. “An architecture for a focused trend parallel web crawler with the application of clickstream analysis”. Inf. Sci., 184(1):, 266–281, 2012. URL http://dblp.uni-trier.de/db/journals/isci/isci184.html#Ahmadi-AbkenariS12.

[11] M. Dikaiakos, A. Stassopoulou, and L. Papageorgiou. “An investigation of web crawler behavior: characterization and metrics”. Computer Communications, 28(8), 880–897, 2005. comcom. 2005. 01.003. URL http://www.sciencedirect.com/science/article/pii/S0140366405000071.

[12] C. Aggarwal, F. Al-Garawi, and P. Yu. “Intelligent crawling on the world wide web with arbitrary predicates”. In Proceedings of the 10th international conference on World Wide Web, WWW ’01, 96–105, 2001.

[13] A. Pal, D. Tomar, and S. Shrivastava. “Effective focused crawling based on content and link structure analysis”. CoRR, abs/0906.5034, 2009. URL http://arxiv.org/abs/0906.5034.

[14] D. Bergmark. “Collection synthesis”. ACM. In Proceedings of the 2Nd ACM/IEEE-CS Joint Conference on Digital Libraries, JCDL ’02, 253–262, New York, NY, USA, 2002. URL http://doi.acm. org/10.1145/544220.544275.

[15] H. Liu and E. Milios. “Probabilistic models for focused web crawling”. Computational Intelligence, 28(3), 289–328, 2012. URL http://dblp.uni-trier.de/db/ journals/ci/ci28.html#LiuM12.

[16] S. Zheng. “Genetic and ant algorithms based focused crawler design. Innovations in Bio-inspired Computing and Applications”, International Conference on, 0, 374–378, 2011. URL: http://doi.ieeecomputersociety.org/10.1109/ IBICA.2011.98.

[17] A. Rungsawang and N. Angkawattanawit. “Learnable topicspecific web crawler”. J. Network and Computer Applications, 28 (2), 97–114, 2005. URL http://dblp.uni-trier.de/db/ journals/jnca/jnca28.html#RungsawangA05

[18] S. Feng, L. Zhang, Y. Xiong, and C. Yao. “Focused crawling using navigational rank”. ACM. In Proceedings of the 19th ACM International Conference on Information and Knowledge Management, CIKM ’10, 1513–1516, New York, NY, USA, 2010. URL http://doi.acm.org/10.1145/1871437.1871660.

[19] C. Manning, P. Raghavan, and H.Schütze. “Introduction to Information Retrieval”. Cambridge University Press, Cambridge, UK, 2008. URL http://nlp.stanford.edu/IR-book/information-retrieval-book.html.

[20] R. Zunino, F. Bisio, C. Peretti, R. Surlinelli, E. Scillia, A. Ottaviano, and F. Sangiacomo. “An analyst-adaptive approach to focused crawlers”. In Advances in Social Networks Analysis and Mining (ASONAM), 2013 IEEE/ACM International Conference on, 1073–1077, Aug 2013.

[21] G. Gouriten, S. Maniu, and P. Senellart. “Scalable,generic, and adaptive systems for focused crawling”. ACM. In Proceedings of the 25th ACM Conference on Hypertext and Social Media, HT ’14, 35–45, New York, NY, USA, 2014. URL http://doi.acm.org/10.1145/2631775.2631795.

[22] G. Gossen, E. Demidova, and T. Risse. “icrawl: Improving the freshness of web collections by integrating social web and focused web crawling”. ACM. In Proceedings of the 15th ACM/IEEECS Joint Conference on Digital Libraries, JCDL ’15, 75–84, New York, NY, USA, 2015. URL http://doi.acm.org/10.1145/2756406.2756925.

الخلاصة

الزاحف المركز مصمم لاسترجاع جزء من الويب لجمع مستندات في موضوع واحد فقط و يهدف الى تعريف الروابط الجيدة التي تقود الى المستندات الهادفة و تجنب التفرعات التي لا تقود للموضوع المطلوب. يوجد عدة دوافع لتصميم الزاحف المركز مثل جلب البيانات ذات العلاقة من الويب وتبسيط فهرسة البيانات, جعل تفاعل المستخدم مع الحاسوب شخصيا و جعل النظام متكيف مع كل مستخدم و كذلك الحاجة الى اداة تغير ستراتيجية البحث لانه ينقح الروابط التي سيتم استرجاعها والحفاظ على حداثة صفحات الويب بالتكيف المستمر و ترشيح الروابط لاهمال الروابط غير المرغوب بها و الحفاظ على تركيز الزاحف بالاعتماد على تفضيل المستخدم. في هذا البحث سوف نشرح طرق لاسترجاع صفحات الويب باستخدام الزاحف التقليدي و الزاحف المركز المقترح. اجرينا العديد من التجارب و اوضحت ان الزاحف المركز المقترح اكثر كفاءة من الزاحف التقليدي في الاسترجاع.