Random Forest Classifier based Scheduler Optimization for Search Engine Web Crawlers

Lakmal Meegahapola, Vijini Mallawaarachchi, Roshan Alwis, Eranga Nimalarathna, Dulani Meedeniya
Department of Computer Science & Engineering
University of Moratuwa, Sri Lanka

Sampath Jayarathna
Department of Computer Science
California State Polytechnic University
California, United States of America

ABSTRACT
The backbone of every search engine is the set of web crawlers, which go through all indexed web pages and update the search indexes with fresh copies, if there are changes. The crawling process provides optimum search results by keeping the indexes refreshed and up to date. This requires an "ideal scheduler" to crawl each web page immediately after a change occurs. Creating an optimum scheduler is possible when the web crawler has information about how often a particular change occurs. This paper discusses a novel methodology to determine the change frequency of a web page using machine learning and server scheduling techniques. The methodology has been evaluated with 3000+ web pages with various changing patterns. The results indicate how Information Access (IA) and Performance Gain (PG) are balanced out to zero in order to create an optimum crawling schedule for search engine indexing.

CCS Concepts
• Information systems → World Wide Web → Web searching and information discovery → Web search engines → Web crawling

Keywords
Web crawler; search engine indexes; optimum scheduler; change frequency; information access; performance gain.

1. INTRODUCTION
Navigating through the different web pages and retrieving the exact content is a difficult and time-consuming task. A search engine is a system that is designed to search the World Wide Web (WWW) for relevant web pages and content [1]. Search engines have made the task of searching the web content easier and less tedious. Two of the major components of a typical search engine are the crawler and indexer, both of which play important roles in the process of providing latest search results [2].

A search engine mainly performs three processes; crawling, indexing and searching in a cyclic manner [3]. Figure 1, depicts the cyclic architecture of a search engine, which facilitates to keep search indexes up-to-date. In other words, the search results of a search engine depend on the search indexes and if the indexes are outdated, the search results may not be up-to-date and accurate.

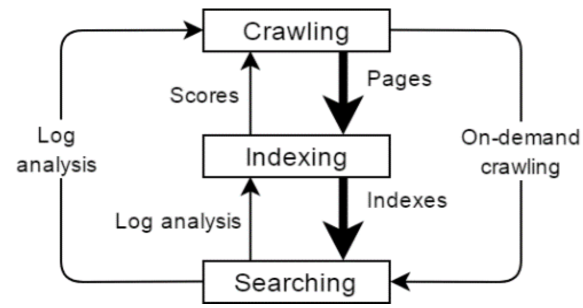

Figure 1: Cyclic architecture of search engines [3]

Web crawlers are dedicated agents in search engines, which crawl the web content and retrieve information for analysis and indexing processes [4]. The crawling process starts from an initial set of URLs named as the *seed URLs*. The crawler visits these seed URLs, identifies the hyperlinks in the web pages and adds them to the list of links to be crawled. URLs from this list are then revisited based on certain policies to refresh the copy of the data available currently. It is crucial to have an optimum schedule for the crawlers to retrieve most recent versions of web pages.

The data obtained from the web crawlers are mainly used by search engines for the process of indexing web pages across the WWW [5] as well as for Change Detection and Notification (CDN) systems [6]. Information in web pages are collected and stored to facilitate easy and accurate retrieval. The behavior of a web crawler is determined by number of special policies [7] to provide information to search engines. However, with the rapid growth of WWW, crawling each web page and retrieving the data to detect changes has become more tedious and expensive [8].

This paper presents a novel approach to determine the change frequency of a web page and optimum crawling schedule to get up-to-date information about web pages to assist in indexing and search engine optimization. Section 2 explores related wok and Section 3 describes the methodology with the frequency detection algorithm. Section 4 discusses the evaluation results and Section 5 concludes the paper stating possible future extensions.

2. BACKGROUND
WWW has developed and expanded to what it is at present, consisting of billions of web pages and documents that keep on ever changing [9]. Search engines have improved the task of browsing and navigating through these web pages in a more user-friendly manner. Processing such large volumes of data has resulted in performance and efficiency issues [10]. Determining

ICSCA 2018, February 8–10, 2018, Kuantan, Malaysia

ACM ISBN 978-1-4503-5414-1/18/02…$15.00
DOI: https://doi.org/10.1145/3185089.3185103

the amount of computational resources that should be assigned to web servers is important [11]. However, it is crucial for search engines to maintain up-to-date indexes of web pages. Many researches have been carried out to increase efficiency of web crawlers. Accurate prediction of significant changes in the content of web pages enables to create an improved incremental crawling strategy that only re-crawls web pages when necessary [12].

A study on scheduling algorithms for web crawling [13] has discussed a major issue in web crawling; the website being crawled can be overloaded, as the crawler can impose a huge load on the web server. Another issue of crawling is that the crawlers have to get updates on web pages on a periodic basis [14], and therefore require a mechanism to detect changes in the web pages which have been already indexed. Here, the authors have suggested an enhanced architecture for the crawlers utilizing client machines, along with a page update algorithm.

Coffman *et al*. [15], have discussed the *binary freshness model*, which can be used to measure the freshness of a web page. It compares the live copy of a particular web page with the cached copy of a web crawler over a specific time period to check if they are identical (or near-identical). However, the binary freshness model lacks the ability to determine whether a page is fresher than the other since the model outputs a binary value; fresh or stale [12]. However, a non-binary freshness model, *temporal freshness metric,* was introduced in [16]. In this model, the longer a cached page remains unsynchronized with its live copy, the more their content tends to drift away from the current page.

Many related studies [17] [18] [19], have identified the Poisson model as a key element to estimate the page changes in a given time interval. Most of the work has assumed that the changes arrive as a Poisson process, and that the average rate of change can be estimated under this model. Grimes *et al.* [18], have described a mechanism to identify changes occurred in a web page and a model to compute the rate of change of a given web site. However, the question of whether all the changes that have occurred are useful to the users is still unanswered.

The study in [20], has proposed new re-crawl scheduling policies that consider the longevity of content in the Web. The authors have stated the crawlers should focus on persistent content and not on transient content such as advertisements. This approach has obtained better freshness at a lower cost and has increased the crawler effectiveness. Further, the work done in [21] states that web crawlers should be aware of web page modifications and has discussed techniques that retrieve information on such modifications. However, the presence of multiple JavaScript and CSS files can reduce the efficiency of certain techniques.

Our previous work. [22], has presented a novel method to detect change frequency in web pages to optimize server-side scheduling of CDN systems using machine learning techniques. Furthermore, in [23], we have described an adaptive technique to carry out web page change detection using multi-threaded crawlers. This method can utilize existing high performance servers in an optimum manner.

Work carried out by Radinsky *et al*. [24], have highlighted the importance of an algorithm to predict changing web content where it is useful in designing a better crawling strategy that only re-crawls pages when it is necessary. The authors have proposed an algorithmic framework for predicting web page changes. It handles temporal features, derived from both single and multiple objects over time. It is claimed that the use of page content and related pages has significantly improved prediction accuracy. However, identifying related pages can be computationally expensive within a general web crawling process.

As discussed, researches have been carried out to determine the change frequency to optimize the crawling process, which is the largest overhead in web page indexing. However, still there is a need for a more efficient, scalable and generalized method for web page change detection while overcoming existing issues.

3. METHODOLOGY

3.1 Change Frequency Estimator

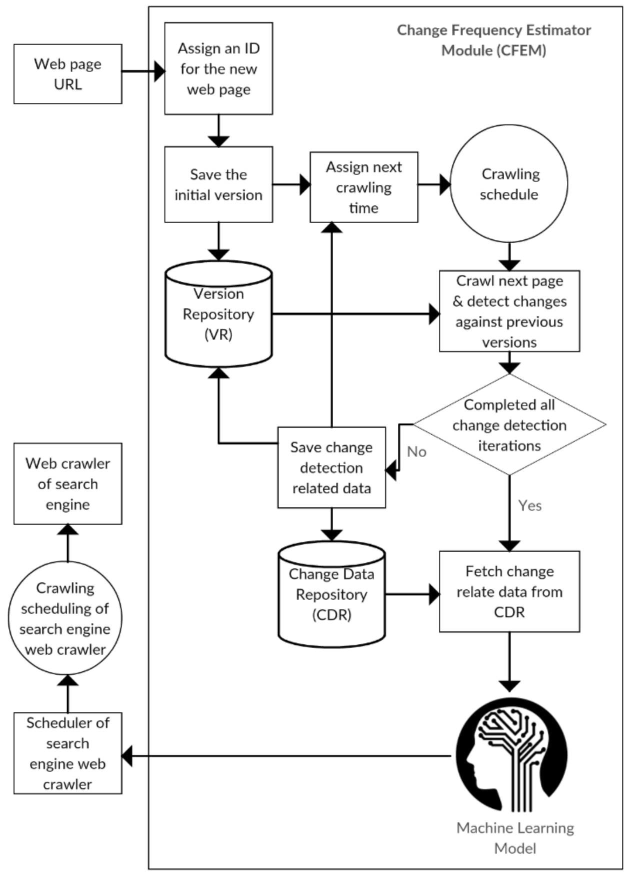

Figure 2. Work Flow of the Change Frequency Estimator Module & its interaction with the Web Search Engine.

The proposed system consists of a list of web pages, where it has already computed the changing frequency (or the refresh rate). When a new page is added to the system, it goes through our module Change Frequency Estimator Module (CFEM). As shown in Figure 2, when the new page is added to the module, it crawls the page and saves the initial version of the web page in the Version Repository (VR). Then it crawls the web page in the specified fixed time intervals: 4 hours, 6 hours, 9 hours, 14 hours, 21 hours, 32 hours, 48 hours and 72 hours, to determine the changes compared to the initial version and subsequent fetched versions of the same web page. The change values for each of the time intervals are then saved in the Change Data Repository (CDR). The change values contain 3 types of content changes (number of new element additions, missing elements, element modifications) and attribute changes.

At the end of all change detection iterations of a web page, data from CDR for that particular web page are sent to the random forest learner model in H2O.ai machine learning API [25], and outputs a loop-time basket. The data from CDR are the features used in the random forest machine learning model described in Figure 3, to predict which loop-time basket a particular web page should be categorized into. In the random forest classifier we used the term *Loop-time basket,* which is a predicted time interval between two crawls for a particular web page.

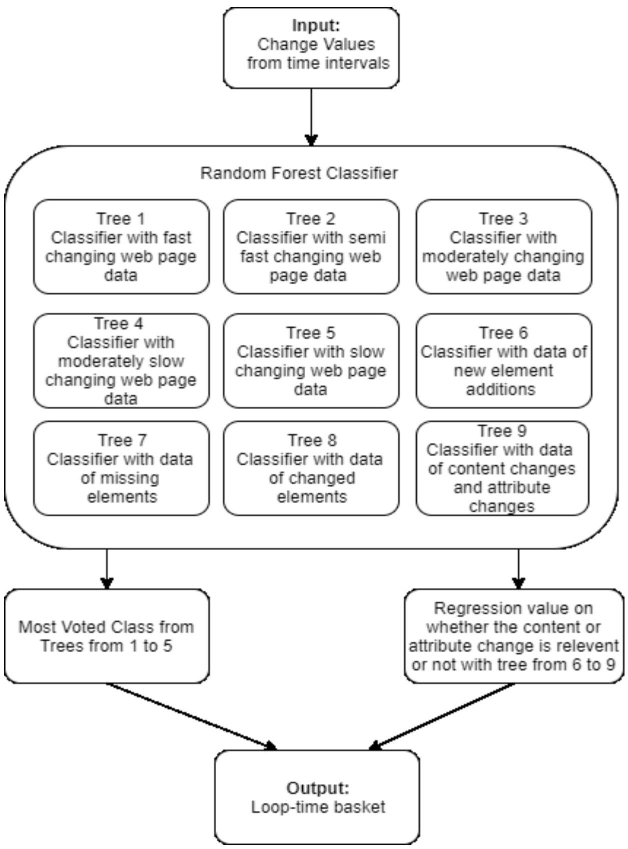

Figure 3: High level view of the Random Forest supervised learning model

The random forest classifier has 9 trees and each tree has a training data set, which is represented using a feature matrix. It should also be noted that each tree contain training data collected from a particular type of web page meaning that data from web pages with a very high change frequency would be present in the feature matrix of tree 1 and data from web pages with a very low change frequency would be present in the feature matrix of tree 5. Each row of the feature matrix represent a set of change values X_4, X_6, X_9, X_{14}, X_{21}, X_{32}, X_{48}, X_{72} of a particular web page where X_4 means the change value after 4 hours from the time that the initial version was stored in VR and X_{72} is the change value after 72 hours from storing the initial version of the web page. Hence, a data point in the matrix can be depicted as given in Equation 1 and n is the number of rows of a feature matrix.

$$\text{Data Point} = x_{i,j} \quad \text{where } (i \in \{1..n\} \text{ and } j \in \{4,6,9,14,21,32,48,72\}) \qquad (1)$$

When a new change value set is sent to the random forest classifier, the change value set would be an array of 8 elements. This value set is passed to each tree in the classifier. The deviation calculation algorithm is used inside the decision trees to find the fit of the newly sent data array with the data set.

$$\text{rowDeviation} = \frac{\sum_i |Xi-Yi|/Xi}{8} \text{ x } 100\% \quad \text{where } i \in \{4,6,9,14,21,32,48,72\} \qquad (2)$$

As given by Equation 1, the algorithm inside the classifier finds the sum of residual errors between each data element of the input array compared to a dataset present in a row of the feature matrix. This rowDeviation calculation suggested by Euation 2 is repeated for each row of the feature matrix as given by Algorithm 1.

The considered Random Forest Classifier has 9 trees, as shown in Figure 3. The first 5 trees combine to give a decision on how often a web page would change and the next 4 trees explore the behavioral elements of the web page in order to predict a regression value (a percentage), whether the web page has relevant changes or irrelevant changes using a modification to the methodology suggested by the work done in [26]. If all the changes are relevant, the regression value would be 100% and if all the changes are not relevant, the value would be 0%. Both regression value and most voted class combine to finally output the predicted loop-time basket for a particular web page as given by algorithm 1.

Algorithm 1 Modified Random Forest Classifier

Require: Array of change values of a web page
Ensure: Find a new perfect frequency to poll the web page

```
input: changeValueArray
treeOutputs = {array with 5 elements initialized to zero}
for each tree in random forest
    threashold1
    votes = 0
    for each row in feature matrix
        rowDeviation = Calculate using equation 2
        if(rowDeviation< threashold1)
            votes++
    treeOutputs.insert(votes/numberOfRows * 100%)
index = indexOfMaximum(treeOutputs)
flag = whether the change is relevant using trees numbered
from 6 to 9
if(flag or index==4)
   loop-time basket = getTheBasket(index)
else
   loop-time basket = getTheBasket(index+1)
output: loop-time basket
```

In this process, each tree checks whether the newly change value data fits the training data set present in the feature matrix of each tree. They output a percentage value describing the fit of the model with the new change value data and the category of the tree with the closest fit is chosen as the output of the classifier given that it is identified as a relevant change using the trees numbered from 6 to 9.

3.2 Experimental Setup

The accuracy of the change detection time-interval classification is tested with a training dataset generated by randomly selecting 80% of the data points from the dataset and the remaining 20% is treated as test data set for each iteration. Hence, for each web page in the test set, we have the loop-time basket which was determined by the machine learning algorithm: say (b1) and the actual loop-time basket, which was determined by the process suggested in [16]: say (b2). We gave a value to the loop-time baskets from 1 to 5, where 1 is the shortest loop-time and 5 is the largest loop-time. For each of the web pages we calculated the difference between b1 and b2. This difference could be considered as the *error* (E) for each web page.

3.3 Data Set

In the current case study, we selected 3122 web pages with various changing patterns, including frequently changing websites to static web pages. In order to train the model in H2O.ai machine learning API based random forest classifier, we collected data from 2498 (3122 x 80%) different web sites over a period of 12 weeks. We divided these 2498 websites into 5 groups based on the frequency in which they change. These web pages were passed through the proposed CFEM and obtained the resultant dataset from CDR for each web page. We used the methodology suggested by [27], to find the refresh rate of web pages. Together with the change values for each of the time intervals, the change frequencies were used for supervised learning model. Further, 10-folds cross-validation is applied for the training data set to avoid the over fitting to a given scenario.

We tested the accuracy of this classifier using another 624 (3122 x 20%) new websites by putting them in one of the 5 loop-time baskets using the changes occurring in each of the time intervals. Therefore, each of the 624 web pages was analyzed using the proposed algorithm and then forwarded to the classifier. We used the method suggested by [27] for all these 624 web pages, to obtain the categorization.

4. RESULTS AND DISCUSSION

Figure 4, shows the learning curve of the supervised classification using H2O.ai machine learning API. The accuracy of the learning algorithm increases with the size of both the training and test datasets. The accuracy can be further improved by having a larger dataset; however, it will increase time complexity due to data labeling. Figure 4 depicts the obtained results and the most accurate results were obtained with the largest data set. The accuracy is measured using the residual error, which is the distance between the actual loop-time basket value and the predicted loop-time basket value.

When analyzing and interpreting the results obtained from the experimental methodology, the two concepts Information Access (IA) and Performance Gain (PG) can be used. IA is a process where no information about successive different versions of the web page is lost due to the small time interval of crawling a particular web page, for which in reality has a larger refresh rate/loop-time. PG is a process where the scheduler does not crawl a web page often in a situation as the web page does not change often. Processing power of the servers is saved because the particular web page is crawled only when needed. As IA and PG have an inverse relationship, the optimum scheduling is done when IA and PG are both 0.

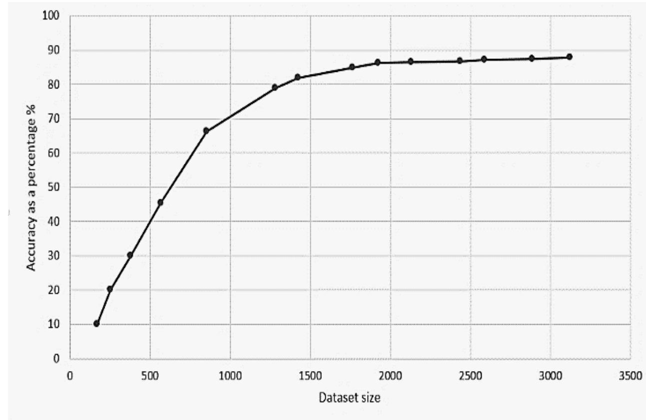

Figure 4: Learning Curve of machine learning model with the size of dataset used for testing and training.

Figure 5, shows the accuracy of the classifier in determining a loop-time basket. Previously we have defined the loop-time basket which was determined by the machine learning algorithm as b1 and the actual loop-time basket which was determined by the process as b2. 87.98% (the precision of the algorithm being 87.98%) of the web pages (549 out of 624) reported an error (E) value of 0; hence, there is no deviation of the CFEM. In 45 cases, we observed an E value of +1, which means, b1 is greater than b2. All these cases have positive IAs because more information is collected than normally needed. However, they have negative PGs because they consume more processing power from the server. 19 cases showed an E of -1, which means, b1 is less than b2. These cases have positive PGs because they are not checked as frequently as they should be. However, they might lead to information losses or negative IAs. Hence, we could say that our machine learning classifier has provided accurate results where most of the E values are 0.

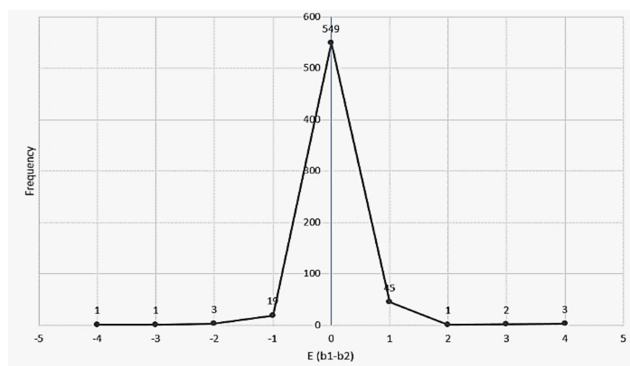

Figure 5: Distribution of E against the frequency/number of web pages for each E

5. CONCLUSIONS

This paper proposes a methodology to detect change frequency of web pages and the evaluation study ascertains the importance of identifying the change frequency of web pages. This helps search engine indexers to efficiently crawl and schedule the indexes up-to-date. In the future, we are interested on extending the proposed work towards distributed servers where frequency detection is based on multiple servers. Further, this research can be extended to increase the accuracy of the machine learning model by increasing the number of features and trees used in creating the random forest classifier.

6. REFERENCES

[1] A. Arasu, J. Cho, H. Garcia-Molina, A. Paepcke and S. Raghavan. Searching the Web. *ACM Transactions on Internet Technology,* 1 (1), 2-43, 2001.

[2] G. Rao, G. Narender, B. Rao and M. Srilatha. Web Search Engine. *International Journal of Scientific & Engineering Research,* 2 (12), 2011.

[3] C. Castillo. 2005. Effective web crawling. *ACM SIGIR Forum,* 39 (1), 55-56, 2005.

[4] K. Desai, V. Devulapalli, S. Agrawal, P. Kathiria and A. Patel. Web Crawler : Review of Different Types of Web Crawler, Its Issues, Applications and Research Opportunities. *International Journal of Advanced Research in Computer Sci,* 8 (3), 1199-1202, 2017.

[5] X. Jin *et al.*, Hybrid Indexing for Versioned Document Search with Cluster-based Retrieval. In *25th ACM International on Conference on Information and Knowledge Management*. ACM, Indiana, USA, 377-386, 2016.

[6] L. B. Meegahapola *et al.*, Change Detection Optimization in Frequently Changing Web Pages. In *Moratuwa Engineering Research Conference (MERCon)*, Moratuwa, Sri Lanka, 111-116, 2017.

[7] S. Mali and B. Meshram. Focused Web Crawler with Page Change Detection Policy. In *2nd International Conference and workshop on Emerging Trends in Technology (ICWET)*. Mumbai, India, 51-57, 2011.

[8] L. Meegahapola et al., Optimizing Change Detection in Distributed Digital Collections: An Architectural Perspective of Change Detection. In *18th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/ Distributed Computing*, Japan, 277-282, 2017.

[9] E. Adar, J. Teevan, S. T. Dumais and J. L. Elsas. The web changes everything: understanding the dynamics of web content. In *2nd ACM International Conference on Web Search and Data Mining*. ACM, Barcelona, Spain, 282-291, 2009.

[10] Y. Shi, L. Wang and F. Du. Performance and energy efficiency of big data systems: characterization, implication and improvement. In *6th International Conference on Software and Computer Applications*. ACM, Bangkok, Thailand, 55-61, 2017.

[11] S. Hagihara, Y. Fushihara, M. Shimakawa, M. Tomoishi and N. Yonezaki. Web server access trend analysis based on the Poisson distribution. In *6th International Conference on Software and Computer Applications*. ACM. Thailand, 256-261, 2017.

[12] C. Olston and M. Najork. Web Crawling. *Foundations and Trends in Information* Retrieval, 4 (3), 175–246, 2010.

[13] C. Castillo, M. Marin, A. Rodriguez and R. Baeza-Yates. Scheduling Algorithms for Web Crawling. In *Proceedings of WebMedia and LA-Web.* IEEE, Ribeirão Preto-SP, Brazil, 10-17, 2004.

[14] D. Yadav, A. Sharma, J. Gupta, N. Garg and A. Mahajan. Architecture for Parallel Crawling and Algorithm for Change Detection in Web Pages. In *10th International Conference on Information Technology*. IEEE, Orissa, India, 258-264, 2007.

[15] E. G. Coffman, Z. Liu and R. R. Weber. Optimal robot scheduling for web search engines. *Journal of Scheduling,* 1 (1), 15-29, 1998.

[16] J. Cho and H. Garcia-Molina. Effective page refresh policies for web. *ACM Transactions on Database Systems,* 28 (4), 390–426, 2003.

[17] D. Fetterly, M. Manasse, M. Najork and J. L. Wien. A large-scale study of the evolution of web pages. *Software: Practice and* Experience, 34(2), 213–237, 2004.

[18] C. Grimes and S. O'Brien. Microscale Evolution of Web Pages. In *17th International World Wide Web Conference*, Beijing, China, 1149-1150, 2008.

[19] D. Ford, C. Grimes and E. Tassone. Keeping a Search Engine Index Fresh: Risk and optimality in estimating refresh rates for web pages. In *40th Symposium on the Interface: Computing Science and Statistics*, Durham, NC, USA, 1-14, 2008.

[20] C. Olston and S. Pandey. Recrawl scheduling based on information longevity. In *17th International World Wide Web Conference*. ACM, Beijing, China, 437-446, 2008.

[21] A. Anjum and A. Anjum. Aiding web crawlers; projecting web page last modification. In *15th International Multitopic Conference (INMIC)*. IEEE, Islsmabad, Pakistan, 245 – 252, 2012.

[22] L. Meegahapola *et al.*, Detection of Change Frequency in Web Pages to Optimize Server-based Scheduling. In *International Conference on Advances in ICT for Emerging Regions (ICTer)*, IEEE explorer, Colombo, Sri Lanka, 165-172, 2017.

[23] L. Meegahapola *et al.*, Adaptive Technique for Web Page Change Detection using Multi-threaded Crawlers. In *7th international conference on Innovative Computing Technology*, London, 2017.

[24] K. Radinsky and P. N. Bennett. Predicting Content Change on the Web. In *6th ACM international conference on Web search and data mining*. ACM. Rome, Italy, 415-424, 2013.

[25] Cook, D. (2016). *Practical Machine Learning with H2O*. 1st ed. O'Reilly Media, Incorporated.

[26] S. Jayarathna and F. Poursardar. Change Detection and Classification of Digital Collections. In *2016 IEEE International Conference on Big Data*. IEEE, Washington D.C., USA. 1750-1759, 2016.

[27] V. M. Prieto, M. A. lvarez, V. Carneiro and F. Cacheda. Distributed and Collaborative Web Change Detection System. *Computer Science and Information Systems,* 12 (1), 91-114, 2015.