An Approach for Web Page Change Detection using Segmentation

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

Master of Engineering
in
Software Engineering

Submitted By
Vishnu Narain Goel
(Roll No. 801231032)

Under the supervision of:
Mr. Vinay Arora
Assistant Professor
Computer Science and Engineering Department

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT
THAPAR UNIVERSITY
PATIALA – 147004

June 2014

Certificate

I hereby certify that the work which is being presented in the thesis entitled, "*An Approach for Web Page Change Detection using Segmentation*", in partial fulfillment of the requirements for the award of degree of Master of Engineering in *Software Engineering* submitted in Computer Science and Engineering Department of Thapar University, Patiala, is an authentic record of my own work carried out under the supervision of *Mr. Vinay Arora* and refers other researcher's work which are duly listed in the reference section.

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

(Vishnu Narain Goel)

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

(Mr. Vinay Arora)
Assistant Professor
Computer Science and Engineering Department

Countersigned by

(Dr. Deepak Garg)
Head
Computer Science and Engineering Department
Thapar University
Patiala

(Dr. S. K. Mohapatra)
Dean (Academic Affairs)
Thapar University
Patiala

Acknowledgement

First of all, I would like to thank the Almighty, who has always guided me to work on the right path of the life. Due to mercy of God, it has been possible for me to reach so far. This work would not have been possible without the encouragement and valuable guidance of my supervisor Mr. Vinay Arora, Assistant Professor, Thapar University, Patiala. I thank my supervisor for his time, patience, discussions and valuable comments. I am equally grateful to Dr. Deepak Garg, Associate Professor and Head, Computer Science and Engineering Department, for motivation and inspiration that triggered me for the thesis work. I am also thankful to the entire faculty and staff members of Computer Science and Engineering Department for their direct-indirect help, cooperation, love and affection, which made my stay at Thapar University memorable.

I will be failing in my duty if I don't express my gratitude to Dr. S. K. Mohapatra, Senior Professor and Dean of Academic Affairs the University, for making provisions of infrastructure such as library facilities, computer labs equipped with net facilities, immensely useful for the learners to equip themselves with the latest in the field.

Vishnu Narain Goel

(801231032)

Abstract

On the basis of changes in web pages, web crawler tends to find, index and re-index the information. Change detection is an activity that aims to identify a particular set of changes according to user specification. The foremost challenge for the web crawler is to detect changes with minimal usage of resources. Continual query and diff algorithm tends to improve the methods for change detection in web crawler. Mild modification in the architecture of web crawler in conjunction with segmentation can enhance its functionality. The main objective of using segmentation in proposed tool is that it offers the customization in change detection technique by selecting particular region of the web page.

In this thesis, a tool named “CFinder” has been proposed, which use diff algorithm in combination with segmentation to identify navigational, structural, and content changes. The proposed tool merged the functionality of generalized with personalized change detection.

Table of Content

List of Figures

List of Tables

Chapter 1
Introduction

Change detection in web crawling includes the detection of changes in a huge collection of decentralized web pages [33]. Web crawlers constantly track the changing web by finding, indexing and re-indexing pages and categorising the changes according to its relevancy.

1.1 Background of change detection

The World Wide Web is expanding in a phenomenal way. The total numbers of web pages are increasing day by day and to index these web pages, web crawler has to implement an efficient and high performance algorithm. The indexed pages need to be updated according to changes happened in the web pages. Moreover, the user has to frequently poll the websites of interest to get newly updated information. According to recent studies 50% web pages remained unchanged during the entire period of time. And for those 50% changed web pages, content change is very less. In the study, only 5% web pages changed very much from their initial version.

The experimental study performed on a large crawl by downloading 151 million HTML pages and computes checksum and figures out rate of change. The degree of change has been classified into different groups such as complete change, large change, medium change, small change, no change. The study figured out that 76% web pages fall in the group of no change and large changed web pages are only 3% and proved that incremental method was efficient in updating web indexes [1].

For change detection in web pages, initially changes are identified in hierarchical structured data and the rate of change was judged by means of experiments, a lot of tools has been developed on the basis of Diff algorithms to locate semantic as well as structural changes such as WEBGUIDE that integrated two existing tools to enable change detection textually and graphically [11]. The AT&T Internet Difference Engine that used push technology to provide customized view of web pages was also based on HtmlDiff algorithm [13]. After that continual queries were used to identify changes. CQ and CONQUER are tools that used the concept of continual query to find changes in the respective web pages. The most important feature of continual

query was pull and push technology so that user don't have to worry about changes as the changes are reflected, system informed the user and user can access particular resource for changed information.

Change detection in web as well as non web is important area of concern as there is continuous need of finding similar pattern and changes in the content of documents. These changes would help in understanding the behaviour of the web and proposed system basically identify content changes by extracting the content part of the web page and then comparing it with previous version and displayed the difference in a more simplified form. But it is a manual change detection system because user has to select both files for comparison. This is one of the major drawbacks of this system. The system should capture bookmarks and download it periodically and find out changes and display it to user as new copy of the web page is opened. For future work, this feature can be incorporated in this system. This system may improve the current browsing experience as currently user has to visit bookmarks regularly to find out the changes.

The previous tools are focusing on user, so far for crawler prospective *i.e.* customization of architecture, new change detection approaches has been identified. WebCQ implemented difference generation, presentation and summarization by means of Continual Query. The first tool that was capable enough to handle static as well as dynamic web pages for change identification [5]. Then change detector, a component based tool that was also a site level monitoring tool for identifying structural as well as content changes provide flexibility by means of graphical user interface and reports so that user can better understand the changes [24].

Table 1.1 Historical background of change detection

Technology	**Year**
Change Frequency Estimation	1997-2004
Continual Query	1998-2004
Diff Approach	1996-2007
Crawler	2002-2013
Segmentation	2011- till now

1.2 Objectives

For a change monitoring system, the main objectives that need to be accomplished are completeness and timeliness [9].

Completeness defined as maximum number of changes needs to be captured and timeliness defined minimizing the delay in capturing changes. For change detection, polling is performed which will tries to figure out changes as soon as possible but for change monitoring system, high frequency polling and processing of pages requires excessive resources.

There is trade off between these two objectives as stock markets require timeliness and web archiving tools demand completeness. According to situation, the objective also varies and is judged with urgency function.

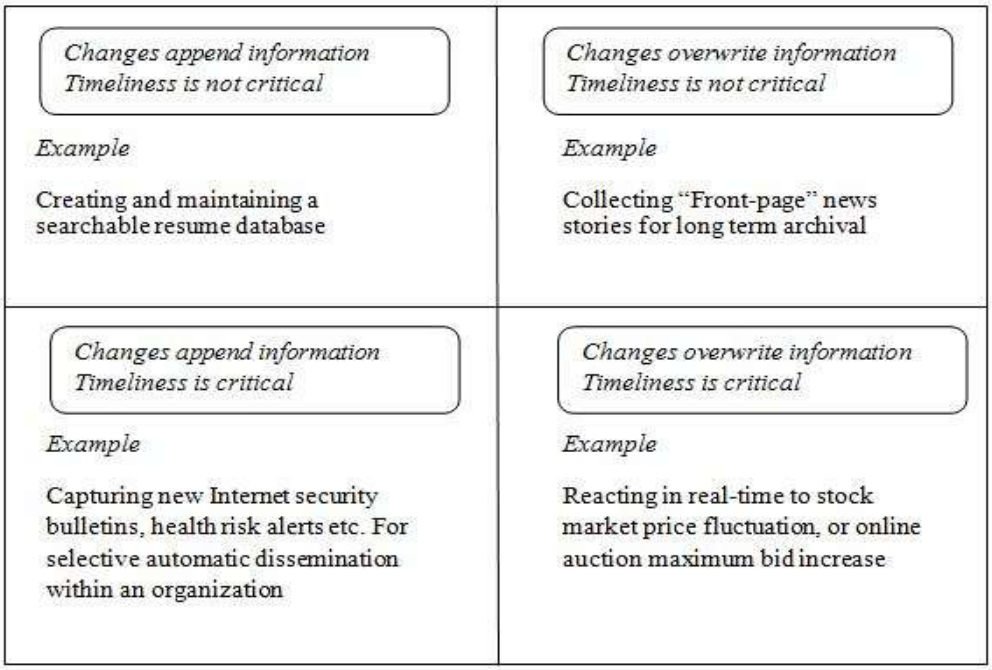


Fig 1.1 Different scenarios with respect to objectives [9]

1.3 Types of change monitoring

Change Monitoring is basically classified into two main categories, assorted monitoring and Linked monitoring.

1.3.1 Assorted monitoring

Users' may be interested in monitoring multiple web pages for changes using a single request or command. For example, a user wants to monitor the contact details of an international company along with some personalized changes such as changes in the discount of a particular product. In this scenario, there are chances that the user might be notified only when both the changes are occurred. These types of request require monitoring multiple web pages with respect to change types. This type of change monitoring is termed as assorted monitoring.

1.3.2 Linked monitoring

The web page can be broadly classified into different sections and with respect to each section; a different page might be linked. In such pages, mainly home page is connected with all respective sections of the web site and if a user wants to monitor the particular section or all the section with respect to change detection, then the page might be divided into frames, and home page can be considered to be a base level with depth 0 and according to that different sections are being leveled for depth calculation. In this scenario, user has to specify the depth of the web site up to which change monitoring is to be performed. For example, in ubuntuforums.org website, mainly different sections correspond to different web pages but all of the sections are linked to home page. Depth is basically a user provided parameter and generally having low value. The low value corresponds not to monitor an entire website.

1.4 Issues related to web crawler

1.4.1 Improving update policy of a data warehouse

A data warehouse stores all the data related to local sites. These types of data are termed as materialized view of the data sources collected autonomously. The data warehouse updates its data during peak hours to attain as maximum updates as possible. But there is problem; if the size of the data became huge then it is difficult to update whole data within the same time period. To solve this, most frequently updated items are identified then the corresponding rows are only updated which may helps in optimizing the utilization of resources as well as the data gets updated.

1.4.2 Improving web caching

A web cache stores recently visited web pages, so that the next time access would not take so much time and can be accessed locally. Caching basically limits the number of accesses to remote resources and reduces access delay and network bandwidth. Generally, most of Web caches are built on LRU (Least Recently Used) page refreshment policy to improve the hit ratio and to better utilize cache. For example, if a web page is inspected daily as compare to one visited each hourly, then the page which is visited daily has less significance and no longer needed in cache as compare to which was visited hourly.

1.4.3 Data mining

In most change detection scenarios, estimation of change frequency can be considered as a good approach for useful information. For instance, when there are unusual changes in all accounts of a bank then bank will take accordingly action. To identify changes more easily, we need to figure out change frequency by repeatedly accessing the same resources such as periodic crawling. Based on repeatedly access, the computation of change frequency for that particular resource is possible. For estimating change frequency there exist a lot of challenges some of them are listed as follows:

1.4.3.1 Incomplete change history

Due to incomplete information of elements change behaviour regarding how often or how much changes are occurred between two accesses, it is quite difficult to predict actual change frequency. Most of web crawler identifies changes after certain interval but it would not explain how many times it is changed in this time period.

1.4.3.2 Irregular access interval

Due to unpredictable nature of user request, management of web cache is quite difficult. In this scenario, system doesn't have any control on how often and when a particular data item is retrieved. The access pattern of information is decided by user in that manner, access interval can be undefined. This irregular access leads difficulty in the estimation of change frequency and hence optimization of the web crawler.

1.4.3.3 Difference in available information

The availability of information varies according to the type of application for different data sources. For example, some of web pages contain information regarding last update and some are without it. Based on this scenario, we need to have different change frequency estimator to identify all aspects of the available information.

1.5 Types of Changes

The change detection system has been used to identify structural and semantic changes in the web documents. Changes are broadly classified into four categories such as content changes, presentation changes, structural changes and behavioral changes.

1.5.1 Content/ Semantic changes

It refers to changes of the page contents from users prospective. For example, a web page created for a weather forecast might be continuously updated on the hourly basis. According to weather respective to city, the page might changes in its presentation to reflect the updates more easily.

1.5.2 Presentation/ Cosmetic changes

These are changes related to the document representation that does not reflect changes in the topic presented in document. For instance, changes to HTML tags can modify the appearance of a Web page while it otherwise remains the same.

1.5.3 Structural changes

It refers to underlying connection of the document to other documents. For example, there are changes in the link destinations of a web directory website. The text/content of the page remains the same but the links pointing to the web pages are modified and it is not reflected in design view of the web page. These changes are very important to detect, as they often might not be visually perceptible.

1.5.4 Behavioral changes

It refers to changes in the active components of a document. For Web pages, this includes scripts, plug-ins and applets. The effects of these types of changes are difficult to predict, as many web pages hide the script code in other files.

1.6 Research Motivation

In most of the Diff algorithms, the changes have been identified by means of creating a Document Object Model Tree that was used by browser for parsing web pages. The parsed page was displayed to user, in the same way these Diff algorithms also create DOM (Document Object Model) tree and then figured out the changes in the web pages. Initially crawler has previous version of the webpage and when the crawler downloads new version of webpage then it compares leaf nodes *i.e.* text nodes on which content is available with new page tree leaf nodes and as soon as changes are identified, it was reflected to user. But this approach was not efficient so to improve the node signature has been introduced that would identify the change from the root. In this approach each node of the tree would have an extra attribute that is a signature which will be unique and for detecting change the signature are being compared and if the root signature is not same in both the cases then page is being modified. But the problem with this approach is that it can be used to identify the change in a whole page, what if the user is interested in a particular part of the web page *i.e.* content or the navigational part that is whether there is some new links are being added or removed from the webpage, then in this case this approach is not so efficient.

To resolve this problem, the web page segmentation came into existence. There are mainly four segmentation methods for the web page [45]. The current approach considered whole page for identification of change that is if the size of page is very big then the respective algorithm would take more time. One of the simple segmentation is dividing the whole page into content and navigational part which is being implemented in the project. One is fixed length page segmentation in this whole page is being divided into various blocks on the basis of some initial size of block. Second is DOM based page segmentation model which is most basic method of segmentation which is being used by web browser. Third is vision based page segmentation, in this approach whole page is divided in the same way user is viewing the web page *i.e.* header, sidebars, footer, content. To identify different blocks, the page can be divided on the basis of pixels that is top 100 pixel would be header and bottom 50 pixel would be footer and in the same way sidebars and the last method of segmentation is Combined/ Hybrid page segmentation which can be implemented by combining any of the previous three approaches to have better results.

CaSePer which was an efficient model for personalized change detection by means of segmentation have implemented DOM based approach with densitometry. The rationale behind the hybrid approach incorporating DOM tree and densitometry was that this approach addressed the segmentation problem by considering the placement of elements in the tree and the density with which the contents are placed [43]. Segmentation not only used in change detection but it was also used in mobile devices so that a particular webpage was displayed on different mobile devices efficiently. CaSePer improved overall complexity that was associated with the change detection process by 77% as compared to previous approaches. This leads segmentation to be a better approach for change detection as compare to Diff algorithms and continual query that didn't considered visual layout of the page and relative importance of different parts of the page.

1.7 Thesis Organization

The Chapter 1 gives the brief introduction about the domain of the topic which includes the background of change detection, objectives such as timeliness and completeness and types of change monitoring system *i.e.* assorted or linked monitoring. The issues related to web crawler and the importance and motivation of change detection. Chapter 2 is the literature review which contains the main research and study part of the thesis; it contains different approaches which are being considered from the very beginning such as by means of databases *i.e.* continual query and diff algorithm, by means of different crawler architecture and by estimating the rate of change and the current approach *i.e.* segmentation for change detection.

Chapter 3 explains the problem statement along with gap analysis. Chapter 4 gives the full implementation which includes proposed algorithm related to problem and the corresponding influence on the web page by means of different phases of HTML code. Chapter 5 includes demonstration of the project work through snapshots along with its description. Lastly the chapter 6 includes the conclusion and the future scope of the research work performed.

Chapter 2

Literature Review

Change detection is one of the most challenging issues for the web as web pages are expanding in exponential rate. Web crawler need to revisit the same set of web pages to identify the semantic as well as syntactic changes. In this literature review, a complete description of change detection method in web from continual query to personalized change detection system is elaborated.

2.1 Using Databases

In databases, a Continual Query (CQ) has played a significant role in the detection of changes. A continual query is a standing query which executes as the specified threshold. A continual query consist of a triple (Q, T_{cq}, Stop), where Q is normal Query (e.g. written in SQL), T_{cq} a trigger condition, and a termination condition Stop. Continual queries can be useful in external applications and as a convenient mechanism for implementing push-based data delivery functions beyond conventional storage, retrieval, and update of data in conventional DBMSs [2]. Example of continual query is as follows

Query: Result = SELECT Student.branch, Student.name FROM Student WHERE Student.rollno = '123';
Trigger: Student.rollno = '123' .and. Student.fee < 'Amount'
Stop: four year from query creation

Calton et al. [2] introduced a distributed event driven continual query system *i.e.* OpenCQ which includes client, server and wrapper. The system was capable of push based content sensitive data management change detection. It has provided a rich event model such as time based or content based triggering mechanism which proved it to be better than CORBA event service. It has also included various continual queries indexing techniques to improve performance and scalability. The architecture of OpenCQ is represented in the Figure 2.1.

In [3], researchers defined a tool named CQ that was based on the concept of

continual queries. These queries are written once and executed over certain data after certain interval of time. If the particular threshold has reached, it informed the user about the change. For each continual query, a robot has been defined which acted as an Intelligent assistant and notified the change to the main program. Wrappers are used because each website has different styles so the way continual queries are being operated may vary so for each website we have different wrapper. It behaved as a translator that converts CQ with respect to website style. It was a personalized change finder that was based on the concept of continual query. It combined the push and pull approach and provided the integrated environment to user.

Conquer, a continual query system introduced by Wei et al. [4] which was used for detecting changes over the web document. That system was capable of handling large number of continual queries by classification in terms of their triggering structure. It has a well defined continual query specification language.

Ling et al. [5, 6] has provided the monitoring and tracking of changes in web pages by means of WebCQ system. The proposed system has personalized notifications and summarization of web pages to users. It also differentiated with other purposed methods with examples. It enlisted the complete mechanism of change detection encountered by WebCQ System. It attained scalability by means of grouping web page sentinel according to their structure.

Gedik et al. [7] has purposed a decentralized self configurable information monitoring system PeerCQ based on continual query. That system enabled better utilization of resources as well as balanced the load between the peers. Each peer has middleware which contained 2 layer, one for peer to peer communication and upper layer for information monitoring such as trigger evaluation, CQ subscription and notifications of change. An effective service partitioning scheme was introduced which integrate heterogeneity of peers with monitoring just like napster and provided flexibility to the system to have as many peers as much it can. The number of nodes in peer to peer system has no limit and whenever a peer was down then its assigned CQ's were transferred to the next peer which was most suitable for it.

The researcher has motivated the use of relational database for detecting content

changes in the ordered XML document. It enabled the change detection for a large document by means of databases. To implement the algorithm, it used SUSXENT schema which was efficient than Xparent schema and applied various queries on leaf nodes to detect the changes [8].

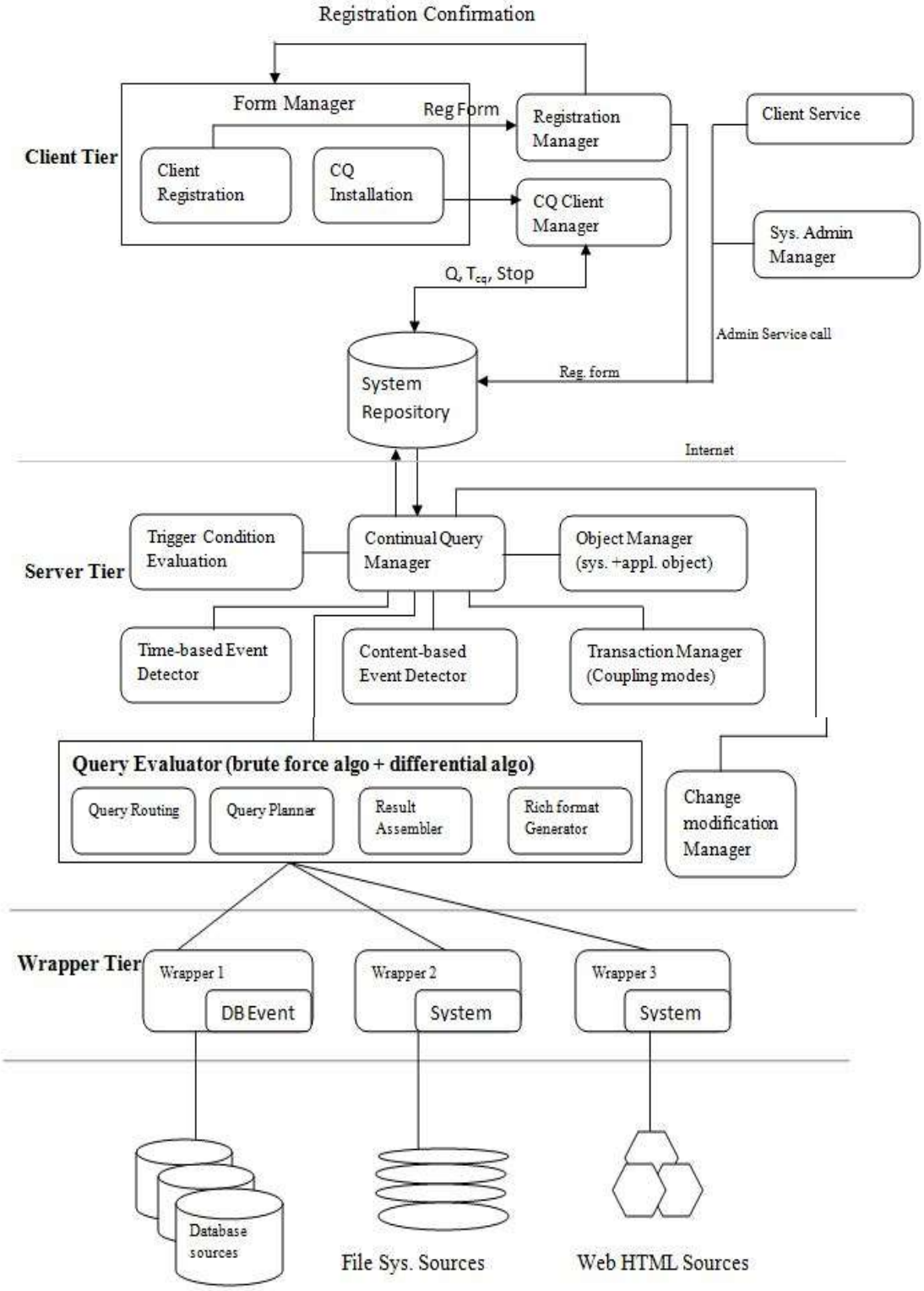


Figure 2.1 Architecture of continual query system [2]

WIC, a new monitoring algorithm proposed by Sandeep et al. [9] which proved to be better approach from other CQ system as it was highly parameterized and there was a balance between timeliness and completeness. It was efficient and could be used in conjunction with other CQ system. The main focus of algorithm was scheduled pulling of web pages.

2.2 Using Diff Algorithm

From the beginning of change detection, a lot of Diff algorithms have been introduced to find the changes in the respective documents. Initially it is implemented on the document level such as LaDiff which was used for detecting the changes in two latex documents. The most common Diff is GNU Diff which uses LCS for detecting changes in two text files.

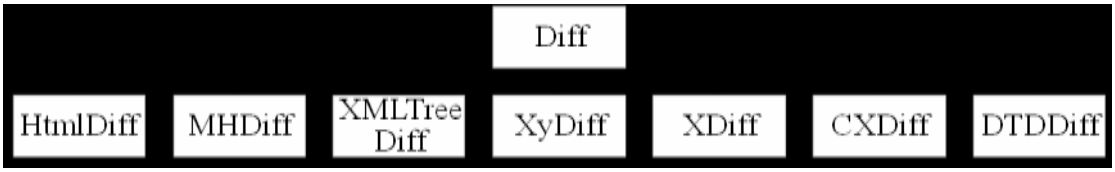


Figure 2.2 Hierarchy of diff algorithm

The researchers has been focused on Hierarchical structured data *i.e.* Latex Document for finding the changes. It divided the whole problem into 2 sub problems *i.e.* Good Matching & Minimum conforming edit script and provided solution of corresponding sub problems with their algorithm and studied these analytically. It purposed LaDiff algorithm for latex documents [10].

Fred et al. [11, 13] has purposed a new system WebGUIDE (Web Graphical User Interface to a Difference Engine) which is combination of two existing tools AIDE, a tool for tracking changes on the web pages and Ciao, a graphical navigator. That system enabled change detection textually and graphically. It also focused on AT&T Internet Difference Engine which provided customized view of web pages for detection of changes. To find different structural changes, it used HtmlDiff along with crawler.

MH-Diff *i.e.* meaningful Diff, a heuristic change detection algorithm derived by Chawathe et al. [12]. This paper has provided higher quality edit script by implementing a large set of operations. The focus of algorithm was structured data which transformed the problem into edge cover set that matched the nodes. Cobena et al. [14] purposed a more efficient and less complex diff approach for xml data warehousing for dynamic as well as static web pages. The purposed approach has focused on more accuracy rather than quality & runs in linear time in average cases. It has tested this approach with syntactic data and found the results close to optimal solution.

The researchers have focused on change detection in web documents by means of WHOWEDA *i.e.* Warehouse of Web Data. Warehouse emphasized web table which was a set of web data and schemas and a set of web algebraic operations such as Web join and web outer join were defined to found changes in those tables [22]. Wang et al. [15] has considered an unordered tree model to be better approach for X-Diff algorithm. It has used node signature and Xhash to find the substantial difference between two xml documents. It improved X-Diff algorithm and still provided near optimal result by means of running time. It has experienced improved X-Diff, X-Diff+ and XyDiff to prove the accuracy of result by improved X-Diff.

Anoop et al. [16] has specified that WebVigiL developed at UT Arlington, a content monitoring system that has the capability of detecting changes in text as well as images but limited in web-based System. This paper focused on extending this system in the context of web by adapting and extending active technology. Chakravarthy et al. [23] has focused on the WebVigiL change detection tool with its specification language and ECA (Event Condition and Action) Paradigm. WebVigiL is a modular system which was customizable according to structure of documents.

CX-Diff purposed by Jacob et al. [17] that has been used for detecting customized changes in content in ordered XML Document in WebVigiL system. WebVigiL can be used on structured as well as semi-structured document and detected changes in XML more accurately. It also used push paradigm to notify user for changes. Chakravarthy et al. [18] has incorporated WebVigiL system architecture to find out various changes in HTML/XML documents. WebVigiL detected changes from a set

of multiple documents by means of CH-Diff and CX-Diff algorithm and provided efficient fetching algorithm such as best effort algorithm. In general, WebVigiL is an automated system for monitoring changes and timely notifications based on user's requirements.

Leonardi et al. [19] has purposed a novel algorithm DTD-Diff to detect structural and semantic changes in the DTD's. The purposed algorithm has improved results over existing algorithm in case of complex DTD's and found to be less expensive as there was no conversion of DTD to XML Schema for change detection and also able to generate optimal or near optimal edit scripts or deltas.

DiffIE, an Internet explorer plugin defined by Teevan et al. [20] which could be used for finding the changes while browsing the internet. For finding the corresponding changes, it caches the page and compares it with previous copy of cache and highlighted the differences. It improved the browsing experience by highlighting the changes in real time.

Saad et al. [21] has specified a new approach for web archiving which used 3 concepts *i.e.* the visual page segmentation, the change detection and importance of blocks in web pages. For change detection, it introduced Vi-Diff algorithm which was much better in extracting changes in visual layout structure of the document. The time of segmentation was much higher than the change detection time.

2.3 Using Crawler

Change detector, a site level web monitoring tool proposed by Boyapati et al. [24] that used machine learning techniques, classification and entity based components for extracting relevant changes from the web pages. The entity based change detection will identify changes in the semantic entities such as price, name or contact number and notify it to the user and then user selects the threshold. In that approach, while crawling web pages the crawler also created site map that would helped in downloading the pages which were in the interest of user rather than whole web site. The GUI of the tool provided flexibility by means of reports to subscriber.

Flesoa et al. [25] motivated the change detection in a selected portion of a webpage. That paper specified a CMW system which enables personalized change notification with query editor that helped in efficient extraction of changes. The specified system helped us to create web update triggers which detected different type of changes in the document. Those web triggers could be used to notify user for respective changes or take respective action according to user request. The CMW system was implemented in java with HTML documents as swing document libraries and it had 2 main component, visual query editor and active query engine that integrated the change detection mechanism. The performance of the system had been evaluated by experimental data and found to be better approach for personalized change detection system. Yadav et al. [26] defined a set of techniques for implementing parallel crawler architecture. It discovered the various issues which might arise while implementing client-server in parallel crawler. In server, it has explained the URL distribution, 2 way communication & priority of linking. It also discussed the client crawler architecture with its basic operations. It mainly concerned about the change detection in page structure, text content and images.

The researchers have focused on text representation of documents for crawling. To index different documents, firstly the document was converted to xml and one of indexing algorithm such as word court algorithm, unique word algorithm, comments algorithm, bold Text algorithm, italic Text algorithm, link algorithm and heading algorithm was applied and after that text search algorithm such as exact query search algorithm or interpreted query search algorithm was applied to find out which indexing and searching algorithm efficiency. Interpreted Query search approach had been used mainly when results from exact query are not satisfactory to the user [27].

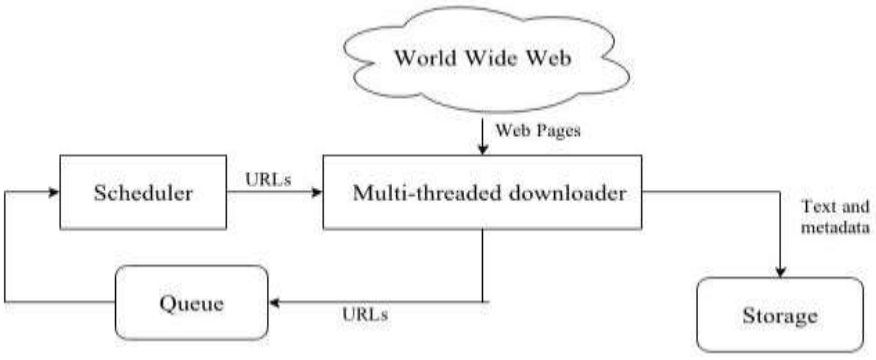


Figure 2.3 General architecture of web crawler [28]

Swati et al. [28] has defined a new approach for the focused web crawler. It proposed page selection and page replacement policy for crawler. It defined 3-step algorithm for page refreshment *i.e.* page relevance computation, determination of page change and update the URL repository. The purposed approach didn't simply download the web page but it firstly find out relevancy of web page and if it was important only then it would download that page so better utilization of resources. It introduced methods for checking the relevancy and find out the structural changes in the page.

Kim et al. [29] has focused on dynamic web crawler implementation in java with the help of tracing hyperlinks in the web pages. It also figured out a new scheduling technique based on current collection cycle time (Ps), Average Cycle time (avg(Ps)) and previous collection cycle (Ps-1).It showed 59% performance benefit compared to static crawling method. Sisi et al. [30] has specified that Surfing notes, an online tool that enabled the user to annotate and archive the webpage for personal use. That tool focused on LCS (Longest Common Subsequence) problem for detecting the text changes and possible solutions are dynamic programming method or dominant matches by row-by-row processing technique. It also performed an experimental analysis on efficient change detection scheduling.

Rawat et al. [31] concerned about focused crawling that used link relevancy (TF-IDF Ranking) criteria for finding the appropriate document. To find out valuable link, it uses best first search which proved to be less resource consumptive as compare to generic & search engine crawler.

2.4 By Estimating Change Frequency

Douglis et al. [32] has figured out the rate of change metric on the basis of live traces from 2 big organization data and found that frequency of change depends mainly on content type and top level domain not on size of the page. It performed semantic comparison on set of documents to find the stable entities such as phone number that would be helpful in detection tools. In that paper, whole trace *i.e.* 950,000 records were analyzed in a statistical way by means of access rate, change ratio, age, modification rate, duplication and graphs to represent variation between changes. That has helped in better prediction of change rate of the web page.

Brewington et al. [33] has provided the mathematical analysis of changes occurring on the web. It figured out the tradeoff in change monitoring of dynamic sources. For estimating the change rate, 200 gigabytes of data is being processed and analyzed on the basis of attributes such as last modified time stamp, remote time stamp and some stylistic information such as links, content or images in the respective web page. The age of web page was represented in the form of poisson process and to formulate the lifespan of the web page. It predicted the current web index on the basis of age and content of source and find out appropriate revisit rate. To identify the current version of the web page in the crawler, β-current is defined which is shown in the Figure 2.4.

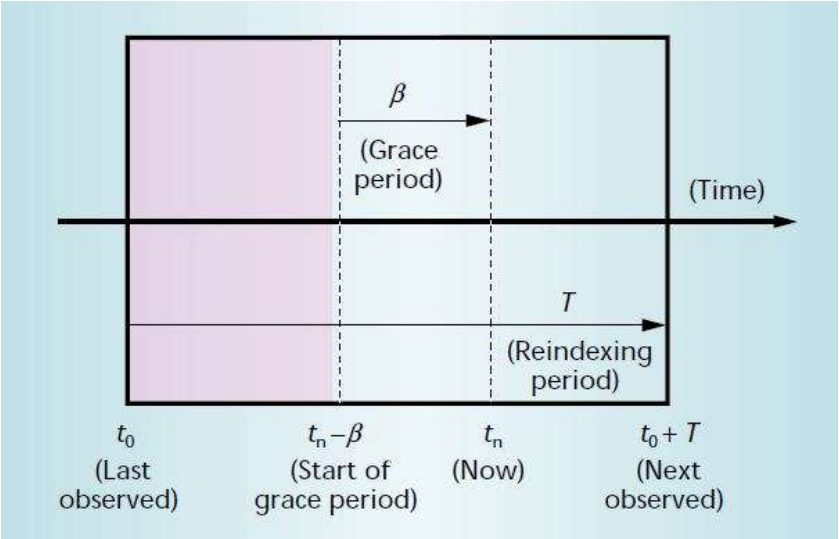


Figure 2.4 Definition of β-current [33]

Cho et al. [34, 35] introduced the concept of estimating the frequency of change for improving the web crawler revisit policy or data ware house update policy where we have incomplete change history or irregular access time span. It purposed various estimators which addresses these problems and find out optimal change frequency on certain conditions & compares it with previous approaches. It also concerned with Walden's Paths Path Manager which was designed to work in distributed environment that extracted the relevant changes based on document signature of linked resources from the web & pass that information to maintainer. It has focused on structural changes rather than presentation changes.

Teevan et al. [36] has conducted an observational study of the problems which peoples are facing while re-finding content in dynamic environment. It considered various forums where re-finding information or problems related to re-finding are

discussed and suggested the solution to be a personalized information management system.

Dalai et al. [37] has focused on Path manager redesigning and suggest context based algorithm, mechanisms to relocate the lost sources and change detection in the web page. The purposed approaches were basically applied on collections rather than path and proved to be better than Walden's Path model.

2.5 Using Segmentation

Segmentation is one of the most exploring areas of research in the field of change detection. In this area, continuous amendments were going on to improve the change detection method. Most of the papers were focusing different aspects of web page segmentation to identify structural and semantic changes that helped crawler to provide updated information to the prospective user.

Chang et al. [38] has specified Graphical change detection. For detecting changes, the whole document has been converted into ordered labeled tree structure based on its markup language and then it compared with the help of tree matching techniques to find out the edit script. The edit script was basically used to identify the change that has been done so far for converting previous page structure to new page structure by means of insert, delete and modify operation. After the change was identified at a node, the user can use any diff algorithm to get more information about the changes. Figure 2.5 shows the basic architecture which is being followed by this paper. It can detect paragraph changes which makes it better approach than Diff algorithm.

Seung et al. [39] has purposed a heuristic semantic change detection (SCD) algorithm for detecting semantic changes in hierarchical structured data. That algorithm figured out changes in the HTML document that contained unstructured information irrespective of XML documents. The considered approach didn’t require any preprocessing or internal structure of the source document that makes it compute frequent changes in any HTML document. To identify changes, both pages are parsed from left to right and top to down manner and its respective trees are constructed and then they are compared. The time complexity of SCD algorithm is O((|X|x|Y|)

log(|X|x|Y|)) where |X| and |Y| is number of branches in syntactic hierarchies in given documents.

Yadav et al. [40] has focused on change detection by HTTP metadata and more efficient change detection algorithm which took linear time as compared to previous algorithms for identifying structural as well as content based changes. The purposed algorithm was divided into 3 steps *i.e.* tree construction, encoding and matching. It has been using tree encoding and level by level tree matching for structural changes. For matching, it has used R.M.S value that was generated from ascii value of the characters in the paragraph. It has implemented algorithm by a specialized set of arrays to represent the relationship between nodes of tree in java.

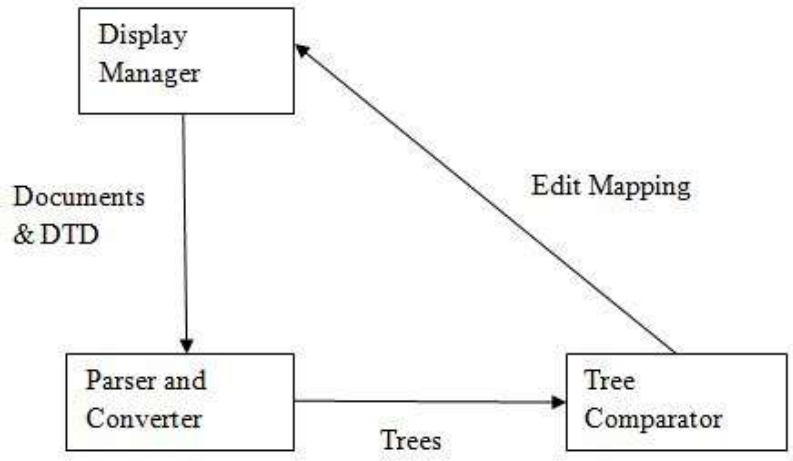


Figure 2.5 Architecture of graphical change detection system [38]

Li et al. [41] purposed a 2 phase algorithm to detect navigation changes in webpage. That paper has considered segmentation of web pages in terms of navigation links and content. The purposed algorithm contain 2 steps, firstly preprocess the web page for removing noisy data such as advertisement, scripts and then block text identification has been carried out. The block text helped in deciding that whether it a navigation page or not. It evaluated DOM Based Identification with AnchorTD or Anchor Text Density and Outlinks/ Pagesize and found to be much more efficient. The purposed algorithm has been experimented on a set of top level domain and showed their results in statistical way. It missed those pages that had one or two small Text blocks and misinterpret the pages.

Law et al. [42] has presented a hybrid web page comparison framework for detecting structural as well as visual changes for web archiving. For change detection only specified region of page was considered rather than whole page. To identify changes, firstly different version of the web pages are captured and each captured version has its own visual and structural descriptor value which was either of whole page or the top of page in terms of BoW (Bag Of Words) representation and then those are compared with new version. To check the performance of the purposed approach, a data set was considered and the results concluded that this approach was more appropriate for detecting visual changes. It also concluded that most relevant changes occur in visible part of web pages without scrolling.

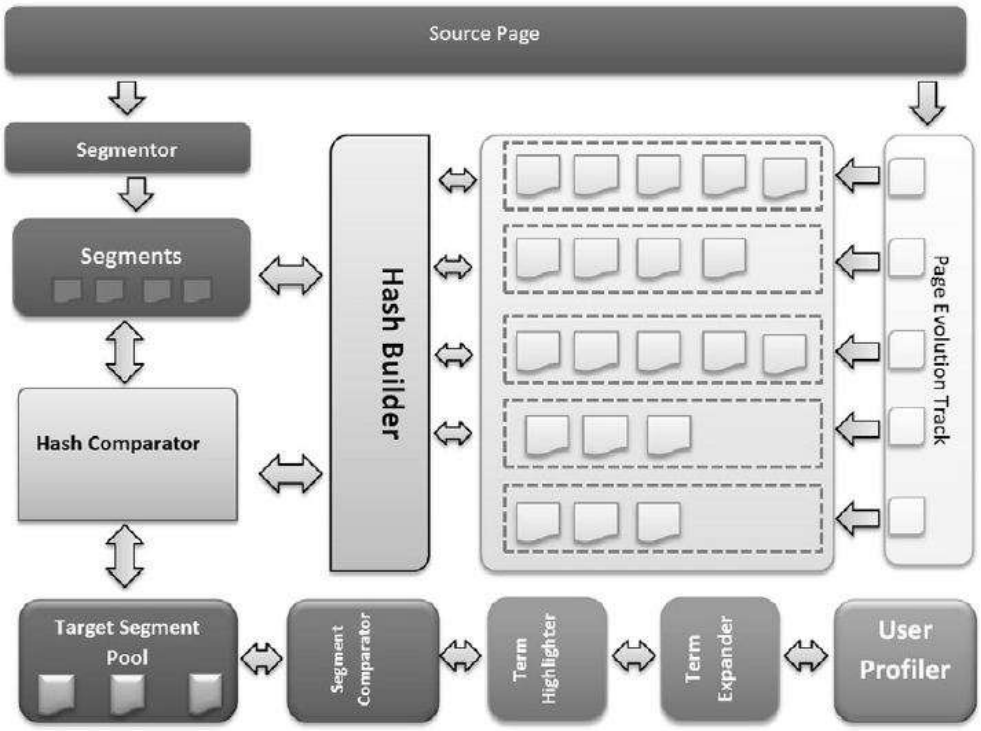


Figure 2.6 Block Diagram of CaSePer Model [43]

Kuppusamy et al. [43] purposed CaSePer, a personalized change detection model that used hybrid page segmentation method *i.e.* DOM tree with densitometry and introduced a ‘‘node boundary and cascaded node sequence’’ based segmentation technique. It has also used MD5 hashing technique to calculate signature and find changes in segments. In the purposed approach, the whole web page is divided into segments and the hash value was computed. That value had been compared with previous version of web page segment and if the value didn’t match then user was notified about the change as shown in figure 2.6. Each user could have its own profile

i.e. they can specify the extent up to which the changes had to be considered. That approach is experimented with a prototype on a number of URL and find out the overall complexity is reduced by 77.8%.

Varshney et al. [44] has provided the detailed and comparative analysis of different algorithm for identifying content and structural change in the web pages. It proposed an efficient algorithm for web page change detection by means of signature and text code. For algorithm, it used document tree based approach. The basic architecture of purposed model is shown in the Figure 2.7. The user can specify the zone in the webpage and according to changes are identified and they were notified to the user.

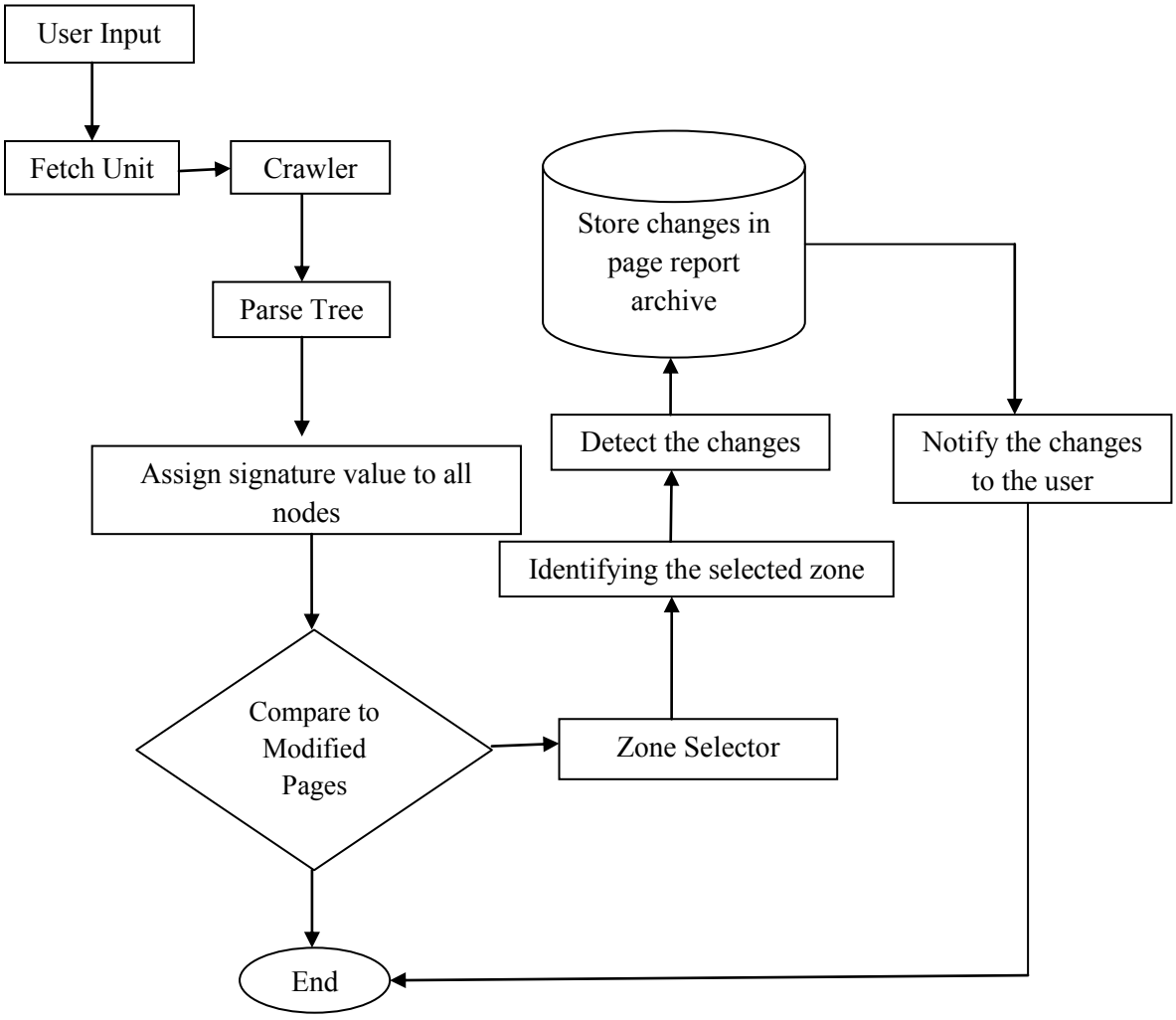


Figure 2.7 Working model of the web page change detection [44]

This is complete description of the related work which is already done in this area and now a table focusing on the main tools which specifically used different approaches to

identify changes such as structural as well as content in the web document is discussed.

Table 2.1 Different tools based on change detection.

Sr. No.	Criteria	Tool Details		
		Tools	**Type of change**	**Concept or algorithm**
1	Databases	OpenCQ[2]	Content	Continual Query
		WebCQ[4]	Content	Continual Query
		PeerCQ[7]	Content	Continual Query
2	Diff Algorithm	WebGuide[11]	Structural	HtmlDiff
		WebVigil[16]	Structural and Content	CH-Diff and CX-Diff
		DiffIE[20]	Content	Cache
3	Crawler	Change Detector[24]	Structural and Content	Entity based
		Surfing Notes[30]	Content	Web page scheduling
4	Segmentation	CaSePer[43]	Structural and content	DOM based segmentation

Chapter 3

Gap Analysis and Problem Statement

Change detection significantly influences the overall performance of web crawler as it invests maximum resources in indexing pages based on changes. The approaches defined for change detection focused on databases, diff algorithm and change frequency estimation.

3.1. Gap Analysis

Based on the literature review of change detection approaches, continual query, diff algorithms and different architecture of web crawlers, following gaps have been identified:

- Till now, no consolidate approach has been developed that extracts or identifies structural changes according to user specification [20, 36, 38].
- No technique that uses segmentation in change detection is available till date that has been combined with existing approaches [15, 26, 28, 42].

3.2. Problem Statement

After reviewing the literature related to search engine, web crawler, change detection techniques and change estimation policies, this has been analyzed that segmentation is one of the major area that can be combined with diff algorithms in web crawler for enhancement related to update policy, web caching and data mining. Segmentation can be defined as a technique in which web page has been divided to get an enhanced perceptive of web page structure.

Among all available diff algorithms, improved XDiff by many researchers find it to be more efficient in terms of complexity, when compared to XDiff+ and XyDiff. A combination of diff with segmentation has been proposed that may lead to better resource utilization for generalized as well as personalized web crawler in terms of results generated based on user specification.

Chapter 4

Proposed Methodology

For the change detection, whole purposed approach is divided into three main sections page cleaning, page segmentation and change detection.

4.1 Proposed CFinder Tool

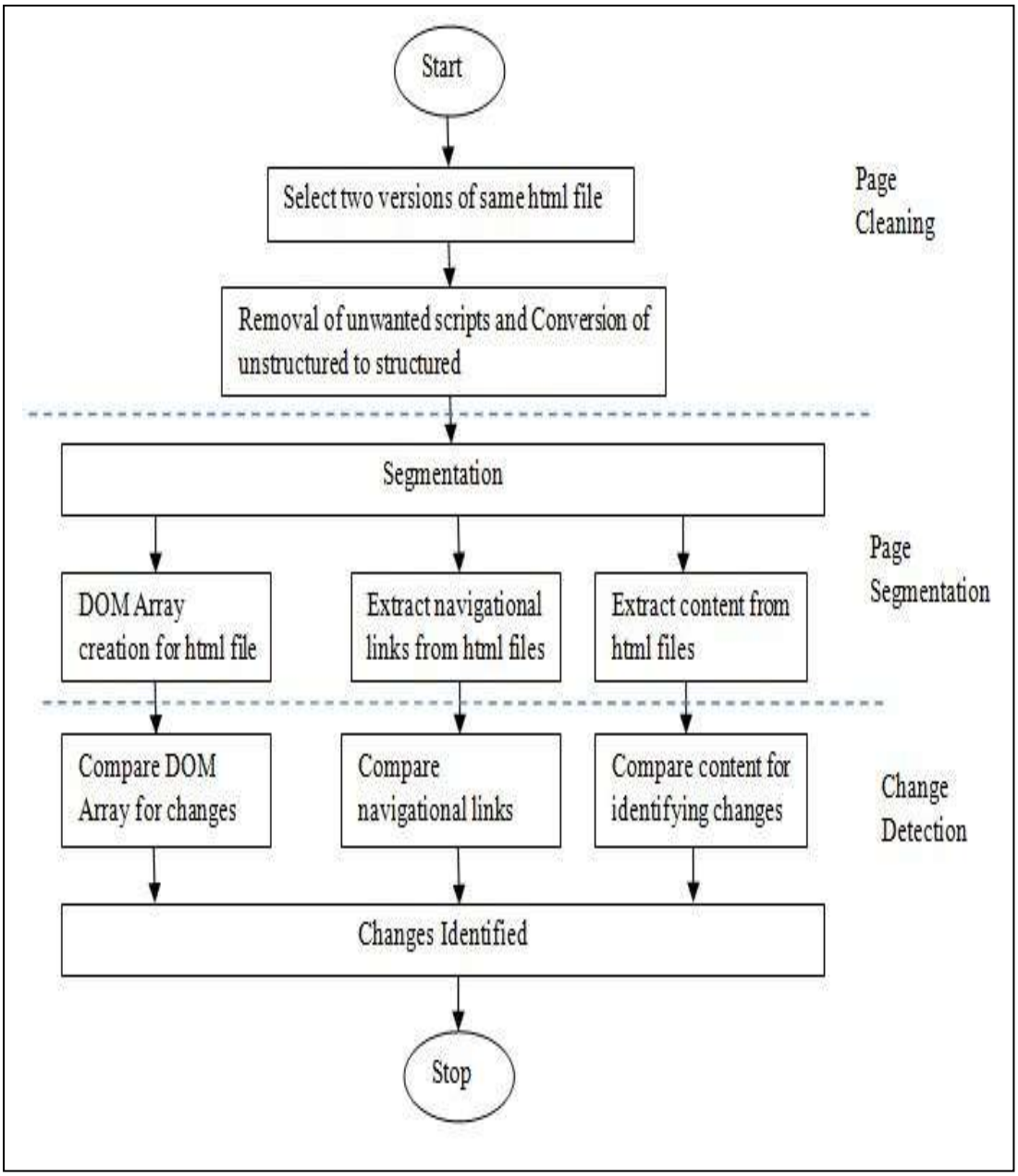


Figure 4.1 Flow chart of CFinder

In change detection, the user can identify changes according to specified requirements *i.e.* selecting a particular region of web page and in previous approaches, the whole

set of changes are being identified and results are notified to the user. In case of structural changes, edit scripts are being generated that would convert one DOM tree *i.e.* source page to another one *i.e.* target page and in case of content changes, sequence alignment algorithm is used.

The purposed methodology is carried out in three steps, page cleaning and page segmentation and change detection. In change detection, sequence alignment algorithm is used for aligning the pattern extracted from the web page.

4.1.1 Page Cleaning

In this phase, input files are selected and page cleaning is done. Page cleaning is required as the most of the web is either ill-formed or unsuitable for processing. To clean up the web page, the order of the tags, attributes and content need to be rearranged. There are a lot of open source HTML parsers which will transforms an dirty HTML web document to clean HTML document for further processing such as HtmlCleaner [47], Neko or Jtidy implemented in Java. They convert an unorganized HTML document into well-formed document which will help in easy segmentation and content extraction.

4.1.2 Page Segmentation

In segmentation, whole web document is divided into three main blocks, a block is referred to as closely related content such as structural block would contain all html tags, attributes and navigational block contains all navigational part of the web page whether it belongs to internal or external and in the same way content block contains all the content part of the web page. The segmentation of web page would result in the DOM array creation for the structural block, list of navigational links for navigational block and extracting the content part of the web page by means of Java API such as Boilerpipe. For source and target page, all the relevant blocks are extracted in the segmentation phase. The flowchart of purposed approach would define all the roles of different phases.

4.1.3 Change Detection

In third phase, change detection is performed by taking input from the segmentation phase. In structural block change detection, DOM array for both web documents are

being aligned with the help of sequence alignment algorithm, the user have privileges to specify the level of the change *i.e.* either up to level 5 or 8 and as all the information in the form of array so it will quite easy to align the sequences and as the algorithm improves the corresponding approach also improves. In navigational change detection, the navigational links are being organized in an array and being aligned in the same way and in the same way for the content blocks.

These are the three phases which will extract the useful information and according to the information retrieved, changes are identified and respective users are notified.

4.2 Proposed Algorithm for Change Detection

4.2.1 Structural Changes

For structural changes, firstly both the web pages are cleaned up by removing scripts or making an unstructured web page to structured web page by means of an existing Java API *i.e.* "HtmlCleaner". After the clean up, respective page is traversed and according to traversal a string which comprises of the initials of the HTML tags and its level in the DOM is created. After that by means of any sequence alignment algorithm both the sequences are aligned and respective changes are being identified.

Algorithm 1: Structural change detection algorithm
Input: Two mutants of HTML file.
Output: Two resultant arrays of the HTML files, aligned sequences for difference
*/*keyword is an array of string containing all the HTML tags, Res Array of result where the required sequences are being stored, i is the index used, K represents initial of keyword being pushed, level is static variable used for tree */*
1. Select two mutants of HTML files
2. For each HTML file
    use HtmlCleaner API *//convert unstructured HTML file to cleaned HTML file*
endFor
3. For each HTML file Read it with bufferedreader line by line
while (buffer != NULL)
    For each line divide it into tokens with StringTokenizer
    If (token contains '<keyword') *//beginning to HTML tag*
        push (keyword)

Res[i] = ‘K’ + level //*Storing level and K in the resultant array*
level++
endif
else
if (token contains '</keyword' && stack[top] == 'keyword')
pop (keyword)
level--
endif
endelse
endFor
endWhile
endFor

4. Use sequence alignment algorithm to identify the changes in the structure of the HTML page

4.2.2 Navigational Changes

In navigational part, whole web page is traversed and the all the anchor tags are being retrieved in top-down fashion and then they are being compared with the new web page anchor tags value if there is change, the changes are being highlighted by means of graphics.

Algorithm 2: Navigational change detection algorithm
Input: Two mutants of same HTML file
Output: A set of links, text content, aligned sequences for difference

1. Select two mutants of HTML files
2. For each HTML file
use HtmlCleaner API *//convert unstructured HTML file to cleaned HTML file*
endFor
3. For each HTML file read it with bufferedreader line by line
While (upto the end of the file)
For each line of the file
If (line contains anchor tag)

retrieve the href value and store it in array

endif

endfor

endWhile

endFor

4. For each HTML files, Different arrays of links will be generated

5. Use sequence alignment algorithm to identify the changes in the navigational part of the HTML pages

4.2.3 Content Changes

In content changes, the respective pages are being traversed and then the text contained in the body tag is retrieved by means of a Java API and the respective text hash sum is computed and then it is compared with the new hash sum if it is not equal then the changes have been occurred.

Algorithm 3: Content change detection algorithm

Input: two mutants of same HTML file

Output: a set of links, text content, aligned sequences for difference

1. Select two versions of same HTML files

2. For each HTML file

use HtmlCleaner API *//convert unstructured HTML file to cleaned HTML file*

endFor

3. For each HTML file

Use Boilerpipe API *//extracts the content from the webpage*

EndFor

4. For each HTML files, extracted content will be generated, now compute hash sum of each file

5. If (hashsum1 != hashsum2)

Print changes occurred

Else

Print No Changes

In Content changes, Boilerpipe or Jsoup API will extract the content by firstly finding the body tag and then retrieving the corresponding information from the body tag

such as text resides in the p tag, pre tag and div tag. This API is widely used for extraction of content from the web pages.

These are the purposed algorithms to identify the different set of changes in the web pages or document. These algorithms have slightly improved performance as compare to previous approaches and as the sequence alignment algorithm improves the overall complexity of the purposed algorithm will also improve.

Chapter 5

Implementation and Results

5.1 Implementation Details

This section covers the details corresponding to implementation of the proposed system.

5.1.1 Technologies used

To implement the proposed system, an application has been developed which incorporates structural, content and navigational change detection. To develop application, Java is used as main programming language on Netbeans IDE.

- Netbeans IDE

It is open source integrated development environment for Java and other programming languages such as PHP, HTML5. It is mainly used for developing application for desktops as well as mobile and licensed under Oracle Corporation. It enables development by means of a set of modular software component. It is designed for windows, mac as well as linux environment.

- Java

Java language offers several features that facilitate with easiness the development and deployment of change detection systems. There are a lot of open source API's which would parse an HTML file and retrieves its information in a particular format. These API depicts the use of java for the development of change detection system. Some of the most common API's are being used for implementing the proposed algorithms which significantly helps in reducing the overall complexity of the system. Java is object oriented programming language which makes it easier to handle HTML tags in a more easy way.

5.2 Experimental Setup

For better understanding of proposed methodology, an example has been discussed that would be explained with the corresponding screen shots that depicts the current state of the algorithm at different phases of the system.

The proposed system has been divided in three phase page cleaning, page segmentation and change detection. In this section, a sample web page has been considered and according to that how the code has been changed in different phases.

In Figure 5.1, the initial version of HTML files that has unstructured and dirty HTML means some unclosed tags and unwanted scripts that may cause problem in parsing the file and retrieving important information from the file. To clean these dirty HTML pages, HTML parsers may be used which remove unwanted scripts and clean up HTML files by structuring the document according to HTML.

```
<html>
  <head>
    <title>
      Scintilla and SciTE
    </title>
  <body bgcolor="#FFFFFF" text="#000000">
    <table bgcolor="#000000" width="100%" cellspacing="0" cellpadding="0" border="0">
      <tr>
        <td width="256">
          <font color="#FFC23F" size="4"> SCINTILLA
        <td width="40%" align="left">
          <font color="#FFCC99" size="4"> :A free source code editor for Win32 and X
        <td width="40%" align="right">
          <font color="#FFCC99" size="3"> Release version 3.4.3<br />
           Site last modified June 11 2014
        <td width="20%">
           
      </tr>
    </table>
    <table bgcolor="#000000" width="100%" cellspacing="0" cellpadding="0" border="0">
      <tr>
        <td width="100%" style="background: url(http://www.scintilla.org/SciBreak2.jpg) no-
repeat;height:150px;">
           
        </td>
      </tr>
    </table>
    <ul id="versionlist">
      <li>Version 3.4.3 fixes hangs and crashes at shutdown on Windows.
      <li>Version 3.4.2 performs word and search match highlighting as an idle task to improve
interactivity
        and allow use of these features on large files.
      <li>Version 3.4.1 fixes a regression in 3.4.0 that stopped the caret moving when lines were wrapped.
      <li>Version 3.4.0 defaults to using strips for find and replace and allows incremental
      searching in the find and replace strips after setting the find.strip.incremental or
```

Figure 5.1 Initial version of HTML file

“Html Cleaner” a Java API is used for cleaning as it contains mainly these features which would differentiate it from other parsers is

- It parses any HTML file or a URL.
- It is thread safe.
- It can be used either from command prompt or from the java code or ant task.

After cleaning, the code is cleaned and then segmentation has been carried out. In segmentation, structural portion of web page has been traversed and according to DOM tree is constructed and according to DOM tree, DOM array is created that contains initial of the tags along with level.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">

<html>
<head>
   <title>Scintilla and SciTE</title>
</head>

<body bgcolor="#FFFFFF" text="#000000">
  <body bgcolor="#FFFFFF" text="#000000">
   <table bgcolor="#000000" width="100%" cellspacing="0" cellpadding="0" border="0">
     <tr>
       <td width="256">
        <font color="#FFC23F" size="4"> SCINTILLA</font>
              </td>
       <td width="40%" align="left">
        <font color="#FFCC99" size="4"> :A free source code editor for Win32 and X</font>
              </td>
       <td width="40%" align="right">
        <font color="#FFCC99" size="3"> Release version 3.4.3<br />
         Site last modified June 11 2014</font>
              </td>
       <td width="20%">
         
            </td>
       </tr>
   </table>
   <table bgcolor="#000000" width="100%" cellspacing="0" cellpadding="0" border="0">
     <tr>
       <td width="100%" style="background: url(http://www.scintilla.org/SciBreak2.jpg) no-
repeat;height:150px;">
         
       </td>
     </tr>
   </table>

```

Figure 5.2 Cleaned version of HTML page

According to DOM tree representation, following array would be generated which would help in detecting changes. Each webpage can be converted to corresponding array which would result in memory storage as page having same structure can point to same array.

The array would be generated looks like this

Page[] ={H0,H1,T2,B1,T2,T3,T4,T5,T5,T5,T5,T2,T3,T4,T5,U2,L3,L3,L3,L3,T2, T3,T2,T3};

Or

Page[] ={Html0,Head1,Title2,Body1,Table2,Tbody3,Tr4,Td5,Td5,Td5,Td5,Table2, Tbody3,Tr4,Td5,Ul2,Li3,Li3,Li3,Li3,Table2,Tbody3,Table2,Tbody3};

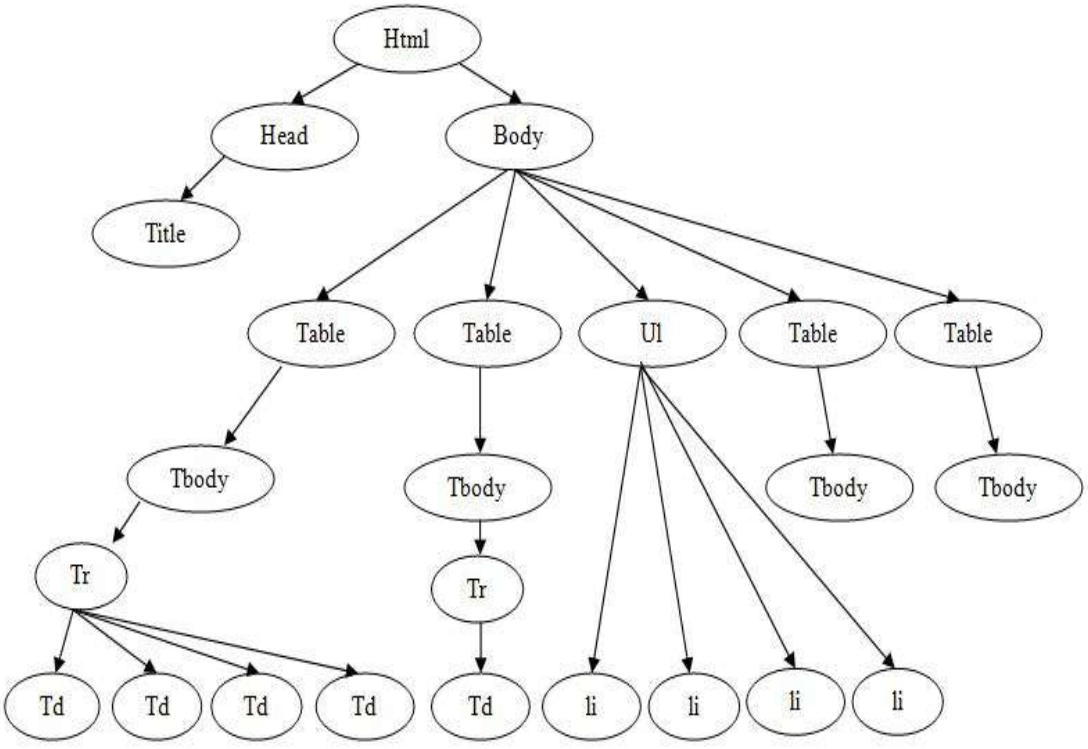


Figure 5.3 DOM tree representation of web page.

After structural code has been generated, the navigational links are extracted and corresponding action has been performed by means of another Java API, Jsoup. It is an open source Java API that makes easier to operate HTML files. The navigational links that has been extracted are compared with the updated page to

identify the updates or change in the links. After navigational changes, content change has been identified and to extract content from the web page same Java API, Jsoup is used.

```
Links: 17
1 http://www.scintilla.org/SciTEImage.html
2 http://www.scintilla.org/SciTEDownload.html
3 http://www.scintilla.org/SciTEDoc.html
4 http://www.scintilla.org/index.html
5 http://code.google.com/p/scite-files/wiki/Customization
6 http://code.google.com/p/scite-files/wiki/Translations
7 http://www.scintilla.org/SciTEFAQ.html
8 http://www.scintilla.org/SciTEDoc.html
9 http://www.scintilla.org
10 http://www.scintilla.org/SciTEImage.html
11 http://www.scintilla.org/SciTEDownload.html
12 http://www.scintilla.org/SciTE-OSX.html
13 http://code.google.com/p/scite-files/wiki/Customization
14 http://groups.google.com/group/scite-interest
15 https://sourceforge.net/project/showfiles.php?group_id=2439
16 https://sourceforge.net/project/?group_id=2439
17 http://sourceforge.net/projects/scintilla
```

Figure 5.4 Navigational section of web page

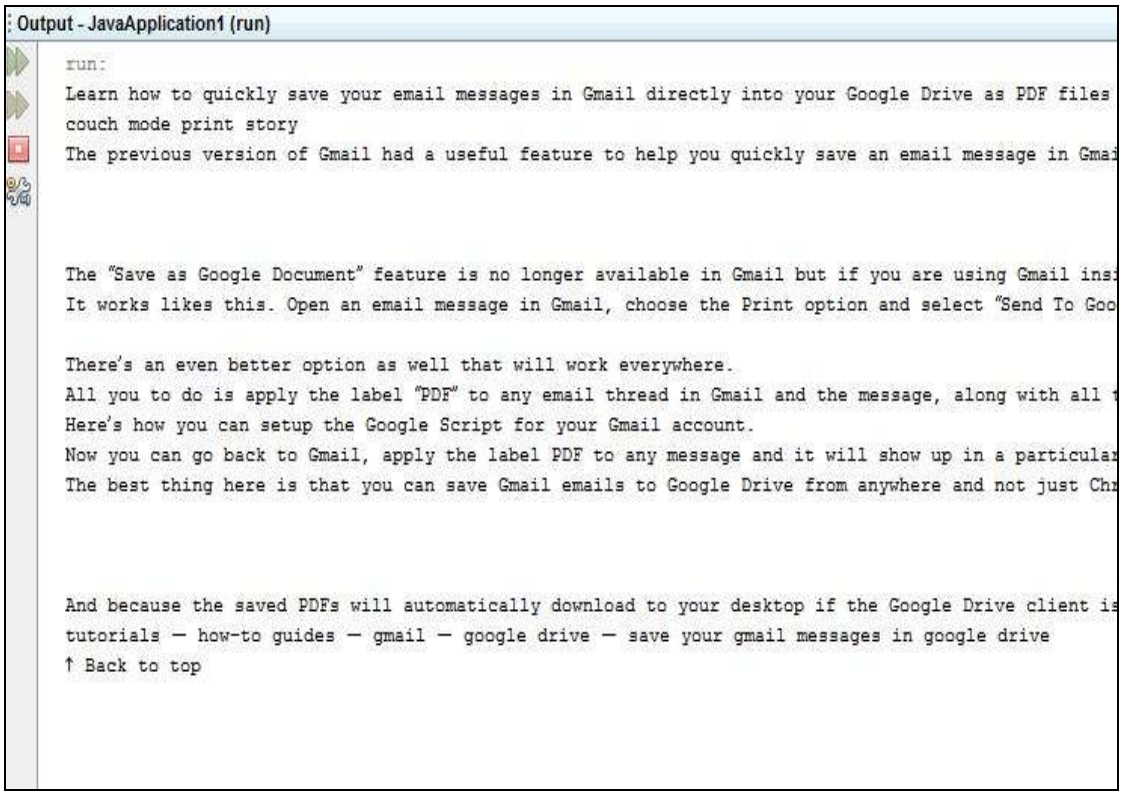


Figure 5.5 Content section of web page

As all the sections of the web pages are extracted, then they are compared with the previous page content, navigation and structural part. According to that changes are find out and notified to user.

5.3 Snapshots of CFinder

The change detection system has been implemented using Java and it is embedded with existing approach such as diff algorithm to check the efficiency. This section covers the detailed screen shots of the change detection system. CFinder has implemented HtmlDiff [46] algorithm and segmentation of web pages in three forms navigational, content and structural. HtmlDiff takes as input two HTML pages and create a new HTML page as output that contain the information about deleted as well as new information.

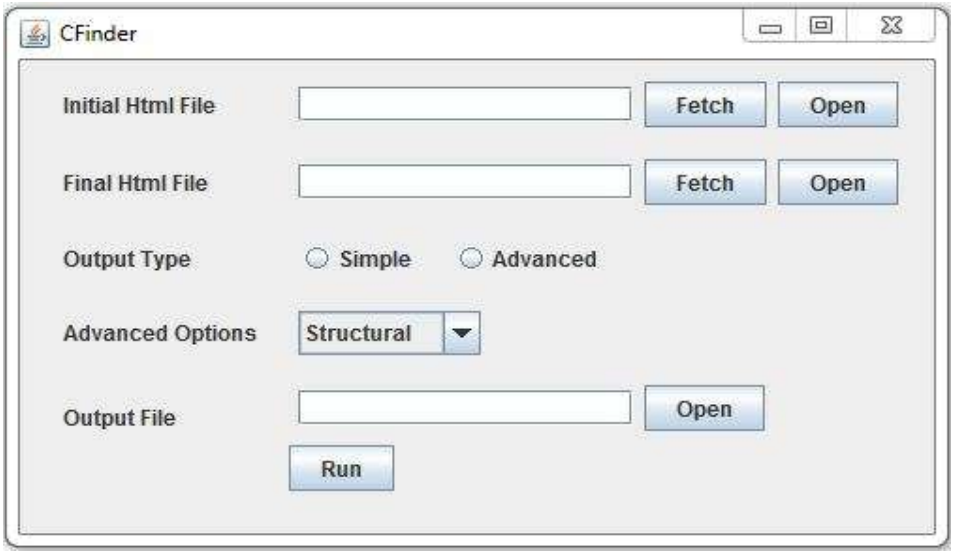


Figure 5.6 GUI for CFinder

This tool integrates the segmentation of web page with the existing Diff approach to improve the overall efficiency of the system. If the client/ user is interested to detect changes in particular part of the page then there is no need to use HtmlDiff algorithm, for detection of change in particular part of page such as navigation & content then the system has to only consider the navigation or content part of the web page. For the input, HTML files can be from the internet or the system, but for the output, file should belong to the system. For selection of the file we would use JFileChoose.

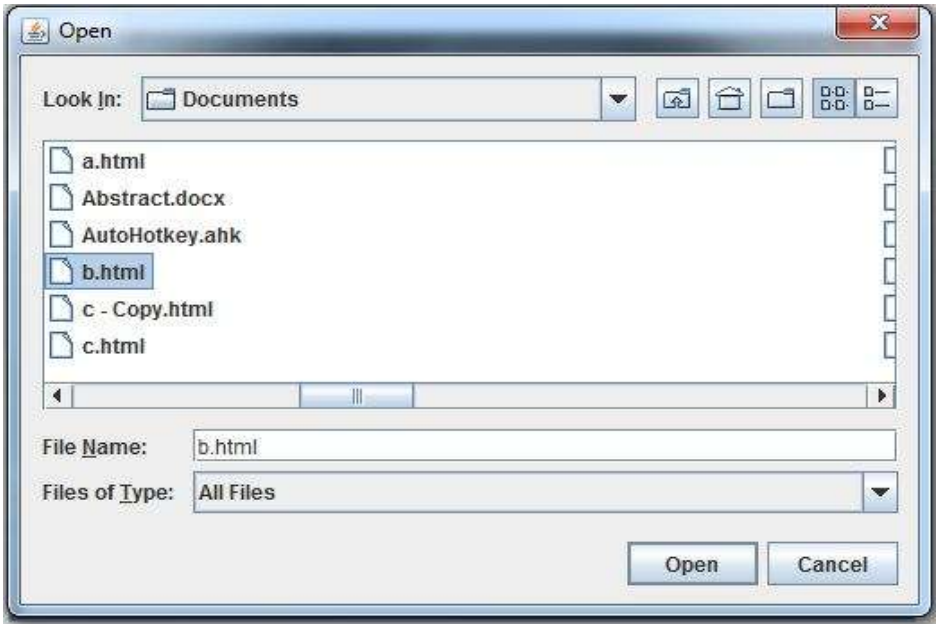


Figure 5.7 File selection window

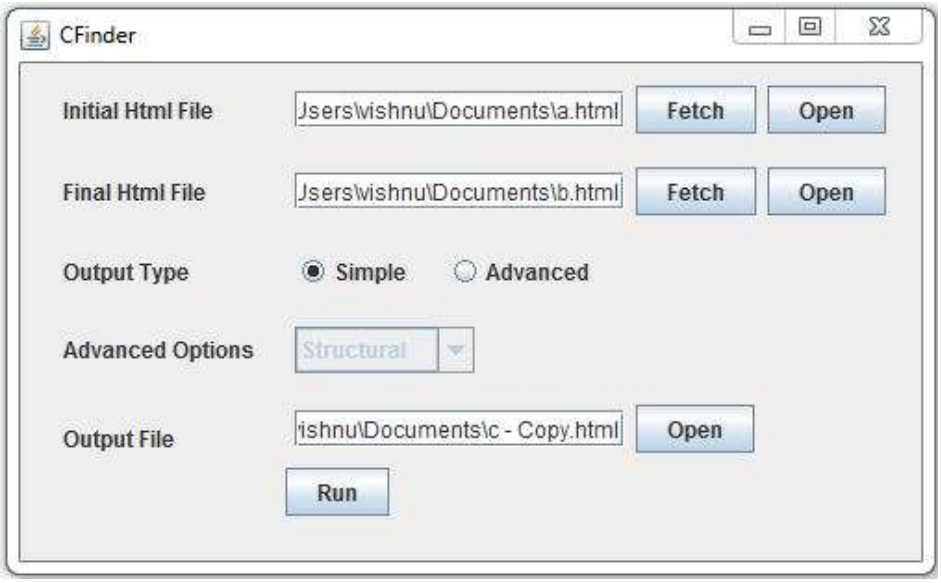


Figure 5.8 Initial inputs

This is developed using java and hence the window you will see for browsing has been developed using swing component of the java. The user can either input a URL from the internet or can specify the local file from the system. If URL is specified, then internet connection is required so that the respective file can be downloaded so that changes can be detected. If the file we have to choose is from the system then we use JFileChooser. JFileChooser is the class which has been used which provides this basic graphical user interface and JFileChooser is an API to show the dialog box

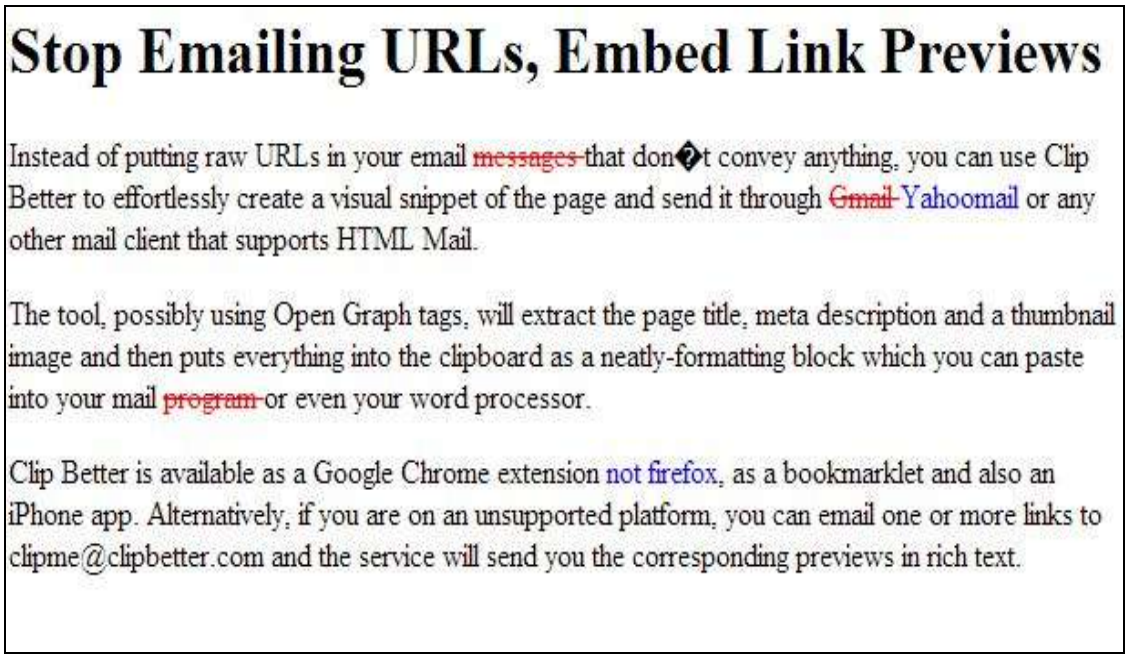
containing file chooser. It also provides various options or you can say event handling on the options that are selected like, if submit is selected or cancel is selected. We can write our business logic on the basis of these decisions. So, when we browse a file and then select, it starts reading that file using File system of the java API.

After selection of files then we would select the output method that is the Simple method which is HtmlDiff approach which will show the output in the specified HTML file. The specified file output would contain both previous as well as new content. And its HTML page would look like Figure 5.9.

```
</div> This is <span class="diff-html-added" id="added-diff-0"
previous="first-diff" changeId="added-diff-0" next="last-diff"
">not </span>amazing
```

Figure 5.9 Code span of output HTML page.

In Figure 5.10, an example of this page is given where the red crossed words are specifying that the respective content is deleted where as blue content is showing that new content or words which are being added. There is customization in the css file of the output page that is when the user can specify colour and formatting of the output window. This feature enable user to have output according to his requirements.



Stop Emailing URLs, Embed Link Previews

Instead of putting raw URLs in your email ~~messages~~ that don't convey anything, you can use Clip Better to effortlessly create a visual snippet of the page and send it through ~~Gmail~~ Yahoomail or any other mail client that supports HTML Mail.

The tool, possibly using Open Graph tags, will extract the page title, meta description and a thumbnail image and then puts everything into the clipboard as a neatly-formatting block which you can paste into your mail ~~program~~ or even your word processor.

Clip Better is available as a Google Chrome extension not firefox, as a bookmarklet and also an iPhone app. Alternatively, if you are on an unsupported platform, you can email one or more links to clipme@clipbetter.com and the service will send you the corresponding previews in rich text.

Figure 5.10 Output HTML file

When we select advanced output method, in that case, we can choose either navigational changes or the content changes.

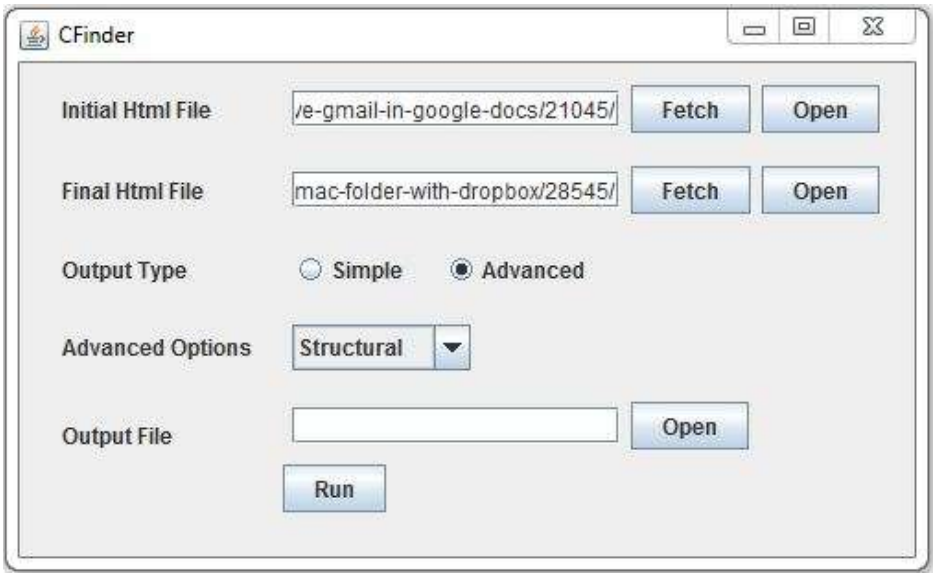


Figure 5.11 Navigation selection window

Figure 5.12 Navigational changes

The navigational change method would retrieve the total number of links which are present in both of the pages. The total number of links is compared and if there is some change in the links then it would retrieve some navigational change otherwise no navigational changes. The deleted links from the previous version of the web page is showed along with newly inserted links.

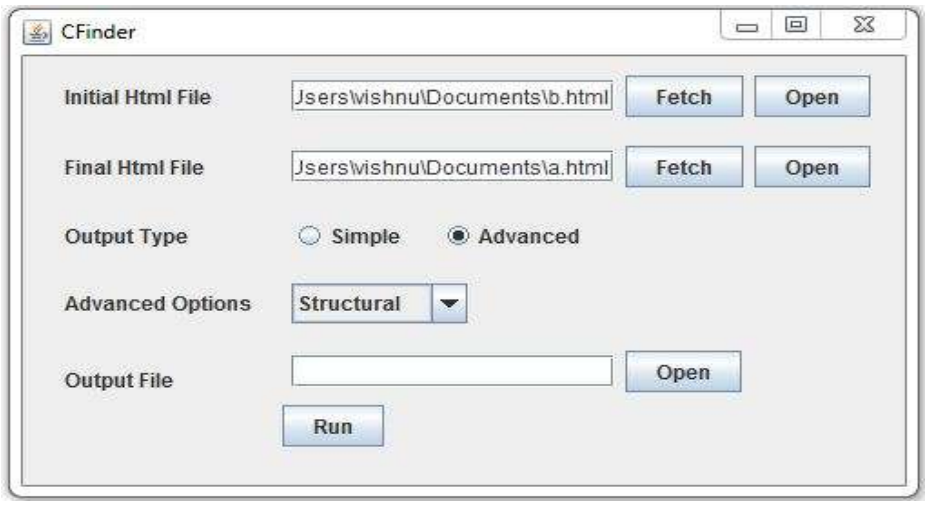


Figure 5.13 Structural selection window

For detecting structural changes, corresponding DOM tree is traversed and according to that DOM array is generated for source page as well as for target page which is represented in Figure 5.14.

```
Source=[0, 1, 1, 2, 2, 2, 2, 0, 0, 0, 0]

Target=[0, 1, 1, 2, 2, 3, 4, 5, 6, 6, 7]
```

Figure 5.14 Structural changes

If you want to view content changes, then you have to select the content option in the advanced output method. These two methods are basically focused on the personalized information change management system. To implement these methods, a HTML parser is being used which would help in retrieving these content and navigational links and then to compare these content longest common subsequence method have to be implemented.

Figure 5.13 and 5.15 is representing the segmentation in the web pages in terms of content and navigation. The content and navigation are two broad areas in which a

page can be segmented. To achieve this, HTML parser by means of jsoup is used which will clean up the HTML files and by cleaning, it mean that HTML is a markup language and there is no compilation of the code so there are chances that some of the tags are unclosed. In the change detection, it is necessary that HTML document should be well formed, all tags must be closed and to do so this parser is used which will give clean HTML file and then we can perform respective action on the HTML files. This parser would help in extracting the content of a particular tag by means of a function which is predefined in this parser. So this parser will help in better retrieval of information from the HTML files.

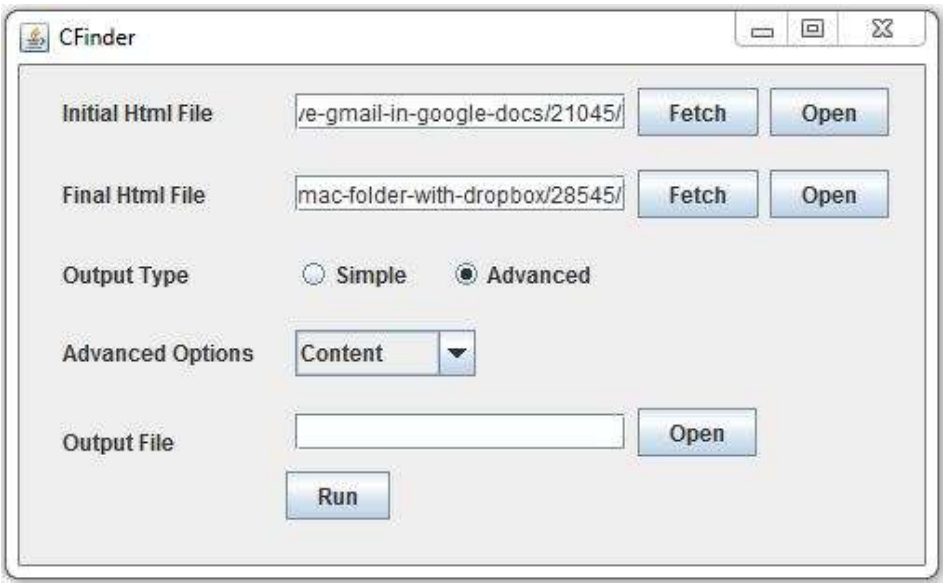


Figure 5.15 Content selection window

```
run:
Content Changed
```

Figure 5.16 Content changes

Chapter 6

Conclusion and Future Scope

6.1 Conclusion

In crawler, change identification and revision of index are the key areas that are needed to be addressed. Diff algorithms, and change in crawler architecture tried to improve the throughput of the crawler by improving the change detection methods. The concept of segmentation in combination with the approaches like diff, customized architecture and change detection estimation may lead to the most suitable crawler for personal and general use. Finding from this thesis are as follow:

- CFinder, a tool based on hybrid approach is proposed.
- The proposed tool integrates the concept of segmentation with Diff approach.
- Diff approach is suitable for generalized change detection, whereas segmentation for personalized change monitoring.

6.2 Future Scope

In future, this approach can be applied for PDF (Portable Document Format), word documents and bit images for identifying the change like semantic and syntactic. The proposed tool, CFinder is based on the concept of simple segmentation that can be extended to DOM based or vision based to improve the overall throughput of the system.

This tool can be automated by means of plug-in that periodically detect changes and send notification to the user. This tool can be merged with other existing techniques like change frequency estimation and customized architecture that may lead to a more efficient generalized change detection system.

References

[1] Dennis Fetterly, Janet Wiener, Mark Manasse and Marc Najork, "A large-scale study of the evolution of web pages", In Proceedings of the 12th International Conference on World Wide Web, ACM, 2003, pp. 669-678.

[2] Calton Pu, Ling Liu and Wei Tang, "Continual queries for internet scale event-driven information delivery", In IEEE Transactions on Knowledge and Data Engineering, vol. 11, no. 4, 1999, pp. 610-628.

[3] Calton Pu, Ling Liu, Wei Tang, David Buttler, John Biggs, Tong Zhou, Benninghoff Paul and Wei Han, "CQ: a personalized update monitoring toolkit", In ACM SIGMOD Record, vol. 27, no. 2, 1998, pp. 547-549.

[4] Ling Liu, Calton Pu, Wei Tang, and Wei Han, "CONQUER: A continual query system for update monitoring in the WWW", In Computer Systems Science and Engineering, vol. 14, no. 2, 1999, pp. 99-112.

[5] Ling Liu, Calton Pu and Wei Tang, "WebCQ-detecting and delivering information changes on the web", In Proceedings of the 9th International Conference on Information and knowledge management, ACM, 2000, pp. 512-519.

[6] Ling Liu, Wei Tang, David Buttler and Calton Pu, "Information monitoring on the web: a scalable solution", In World Wide Web, vol. 5, no. 4, 2002, pp. 263-304.

[7] B. Gedik and Ling Liu, "PeerCQ: a decentralized and self-configuring peer-to-peer information monitoring system", In Proceedings of 23rd International Conference on Distributed Computing Systems, 19-22 May 2003, pp.490-499.

[8] Erwin Leonardi, S.S. Bhowmick, T.S. Dharma and Sanjay Madria, "Detecting content changes on ordered XML documents using relational databases", In Database and Expert Systems Applications, Springer Berlin Heidelberg, 2004.

[9] Sandeep Pandey, Kedar Dhamdhere and Christopher Olston, "WIC: A general-purpose algorithm for monitoring web information sources", In

Proceedings of the 30th International Conference on Very large databases (VLDB), 2004, pp. 360-371.

[10] Sudarshan S. Chawathe, Anand Rajaraman, Hector Garcia-Molina and Jennifer Widom, "Change detection in hierarchically structured information", In ACM SIGMOD Record, vol. 25, no. 2, ACM, 1996, pp. 493-504.

[11] Fred Douglis, Thomas Ball, Yih-Farn Chen and Eleftherios Koutsofios, "WebGUIDE: Querying and navigating changes in web repositories", In Computer Networks and ISDN Systems, vol. 28, no. 7, 1996, pp 1335-1344.

[12] Sudarshan S. Chawathe, and Hector Garcia-Molina, "Meaningful change detection in structured data.", In ACM SIGMOD Record, vol. 26, No. 2, 1997.

[13] Fred Douglis, Thomas Ball, Chen Yih-Farn, and Eleftherios Koutsofios, "The AT&T Internet Difference Engine: Tracking and viewing changes on the web", In World Wide Web, vol. 1, no. 1, 1998, pp. 27-44.

[14] Gregory Cobena, Serge Abiteboul and Amelie Marian, "Detecting changes in XML documents", In Proceedings of 18th IEEE International Conference on Data Engineering, 2002, pp. 41-52.

[15] Y. Wang, D.J. DeWitt, J.Y. Cai, "X-Diff: an effective change detection algorithm for XML documents", In Proceedings of 19th International Conference on Data Engineering, 5-8 March 2003, pp.519-530.

[16] Anoop Sanka, Chamakura Shravan and Chakravarthy Sharma, "A dataflow approach to efficient change detection of HTML/XML documents in WebVigiL", In Computer Networks, vol. 50, no. 10, 2006, pp.1547-1563.

[17] Jyoti Jacob, Alpa Sachde and Chakravarthy Sharma, "CX-DIFF: a change detection algorithm for XML content and change visualization for WebVigiL", In Data & Knowledge Engineering, vol. 52, no. 2, 2005, pp.209-230.

[18] S. Chakravarthy and S.C.H. Hara, "Automating Change Detection and Notification of Web Pages (Invited Paper)", In 17th International Workshop on Database and Expert Systems Applications (DEXA), 2006, pp.465-469.

[19] Erwin Leonardi, T. Hoai Tran, Sourav S. Bhowmick and Sanjay Madria, "DTD-Diff: A change detection algorithm for DTDs", In Data & Knowledge Engineering, vol. 61, no. 2, 2007, pp.384-402.

[20] Jaime Teevan, Susan T. Dumais, Daniel J. Liebling, and Richard L. Hughes, "Changing how people view changes on the web", In Proceedings of the 22nd annual ACM symposium on User interface software and technology, ACM, 2009, pp.237-246.

[21] Myriam Ben Saad and Stéphane Gançarski, "Using visual pages analysis for optimizing web archiving", In EDBT/ICDT Workshops, ACM, 2010, pp.43.

[22] S.S. Bhowmick, S.K. Madria, Wee-Keong Ng, "Detecting and representing relevant Web deltas in WHOWEDA", In IEEE Transactions on Knowledge and Data Engineering, vol. 15, no.2, March-April 2003, pp.423-441.

[23] Chakravarthy Sharma, Anoop Sanka, Jyoti Jacob and Naveen Pandrangi, "A learning-based approach for fetching pages in webvigil", In ACM symposium on Applied computing, 2004, pp.1725-1731.

[24] Vijay Boyapati, Kristie Chevrier, Avi Finkel, Natalie Glance, Tom Pierce, Robert Stockton, and Chip Whitmer, "ChangeDetector™: a site-level monitoring tool for the WWW", In Proceedings of the 11th International Conference on World Wide Web, ACM, 2002, pp.570-579.

[25] Sergio Flesca and Masciari Elio, "Efficient and effective web change detection", In Data & Knowledge Engineering, vol. 46, no. 2, 2003, pp. 203-224.

[26] Divakar Yadav, A. K. Sharma and J. P. Gupta, "Parallel crawler architecture and web page change detection", In WSEAS Transactions on Computers, vol. 7, no. 7, 2008, pp. 929-940.

[27] D. Minnie and S. Srinivasan, "Intelligent Search Engine algorithms on indexing and searching of text documents using text representation", In Proceeding of International Conference on Recent Trends in Information Systems (ReTIS), 21-23 Dec. 2011, pp.121-125.

[28] Swati Mali and B. B. Meshram, "Focused Web Crawler with Page Change Detection Policy", In International Journal of Computer Applications, 2011, pp.51-57.

[29] K. S. Kim, K. Y. Kim, K. H. Lee, T. K. Kim, W. S. Cho, "Design and implementation of web crawler based on dynamic web collection cycle," In Proceeding of International Conference on Information Networking (ICOIN), 1-3 Feb. 2012, pp.562-566.

[30] He Sisi, E. Chan, "Surfing Notes: An Integrated Web Annotation and Archiving Tool", In Proceeding of IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology (WI-IAT), vol. 3, 4-7 Dec. 2012, pp.301-305.

[31] S. Rawat and D.R. Patil, "Efficient focused crawling based on best first search", In Proceeding of IEEE 3rd International on Advance Computing Conference (IACC), 22-23 Feb. 2013, pp.908-911.

[32] Fred Douglis, Anja Feldmann, Balachander Krishnamurthy and Jeffrey C. Mogul, "Rate of Change and other Metrics: a Live Study of the World Wide Web", In USENIX Symposium on Internet Technologies and Systems, vol. 119, 1997.

[33] B.E. Brewington and G. Cybenko, "Keeping up with the changing Web", In Computer, vol. 33, no.5, May 2000, pp.52-58.

[34] Cho, Junghoo and Hector Garcia-Molina, "The evolution of the web and implications for an incremental crawler", In Proceedings of the 26th International Conference on Very Large Data Bases (VLDB), USA, 1999, pp. 200-209.

[35] Cho, Junghoo, and Hector Garcia-Molina, "Estimating frequency of change", In ACM Transactions on Internet Technology (TOIT), vol. 3, no. 3, 2003, pp. 256-290.

[36] J. Teevan, "How people re-find information when the web changes", In MIT AI memo 2004-12, June 2004.

[37] Z. Dalai, S. Dash, P. Dave, L. Francisco-Revilla, R. Furuta, U. Karadkar, F. Shipma, "Managing distributed collections: evaluating Web page changes, movement, and replacement", In Proceedings of Joint ACM/IEEE Conference on Digital Libraries, 7-11 June 2004, pp.160-168.

[38] G.J.S. Chang, G. Patel, L. Relihan, J.T.L. Wang, "A graphical environment for change detection in structured documents", In Proceedings of The Twenty-First Annual International Computer Software and Applications Conference, (COMPSAC), 11-15 Aug 1997, pp. 536-541.

[39] Seung-Jin Lim, Yiu-Kai Ng, "An automated change-detection algorithm for HTML documents based on semantic hierarchies", In Proceedings of 17th International Conference Data Engineering, 2001, pp.303-312.

[40] D. Yadav, A.K. Sharma, J. P. Gupta, "Change Detection in Web Pages", In Proceedings of 10th International Conference on Information Technology, (ICIT 2007), 17-20 Dec. 2007, pp.265-270.

[41] Li Yue, Dong Shou-bin, Zheng Xiang, Ma Bin-Hua, "Improving navigation page detection by using DOM-based block text identification", In Proceedings of 10th International Conference on ICT and Knowledge Engineering (ICT & Knowledge Engineering), 21-23 Nov. 2012, pp.129-134.

[42] Law, Marc Teva, Nicolas Thome, Stéphane Gançarski, and Matthieu Cord, "Structural and visual comparisons for web page archiving", In Proceedings of the ACM symposium on Document engineering, ACM, 2012, pp. 117-120.

[43] K.S. Kuppusamy and G. Aghila., "CaSePer: An Efficient Model for Personalized Web Page Change Detection Based on Segmentation", In Journal of King Saud University Computer and Information Sciences, 2013.

[44] N.K. Varshney, D.K. Sharma, "A novel architecture and algorithm for web page change detection", In Proceedings of IEEE 3rd International Advance Computing Conference (IACC), 22-23 Feb. 2013, pp.782-787.

[45] Yesilada, Yeliz, “Web Page Segmentation: A Review”, Technical Report, University of Manchester and Middle East Technical University Northern Cyprus Campus, 2011.

[46] HtmlDiff API [online]. Available: https://code.google.com/p/daisydiff/ , 2014.

[47] HtmlCleaner API [online]. Available: http://htmlcleaner.sourceforge.net/, 2014.

List of Publication

Published/ Accepted

[1] Vishnu Goel and Vinay Arora, "A Survey Paper on Web Page Change Detection" in International Conference on Mathematics and Engineering Science, Engineering Sciences International Journal, Vol.2, Chitkara University, Chandigarh, March 2014, pp. 241-245 [Published].

[2] Vishnu Goel and Vinay Arora, "A Hybrid Approach for Web Page Change Detection" in 1st International Conference on Networks & Soft Computing (ICNSC), Vignan's University, Guntur, Andhra Pradesh, 19-20 August 2014 [Accepted].

Communicated

[1] Vishnu Goel and Vinay Arora, "A Novel Approach for Web Page Change Detection based on Segmentation" in International Conference on Soft Computing & Machine Intelligence (ISCMI), New Delhi, 26-27 September 2014.