International Journal of Computer Applications Technology and Research
Volume 3– Issue 4, 254 - 262, 2014, ISSN: 2319–8656

A Web Page Change Detection System For Selected Zone Using Tree Comparison Technique

Sandesh D. Jain
Computer Science and Technology
Department of Technology
Shivaji University
Kolhapur, Maharashtra
India

H.P.Khandagale
Computer Science and Technology
Department of Technology
Shivaji University
Kolhapur, Maharashtra
India

Abstract: This paper describes Web Page Change Detection System for Selected Zone based on tree comparison mechanism corresponding to HTML pages. Two sub trees for the selected zone will be generated one for initial and another for changed version of Web Document. The Generalized Tree Comparison Algorithm is developed to compare these sub trees for selected zone. This algorithm uses the properties of HTML page and heuristics as a node of the trees. This proposed system will include functionalities and interfaces for processing user request, fetching web pages from the internet allowing users to select zone in web pages to monitor. This method performs well and is able to detect the structural as well as content level changes even at the minute level and helps to locate minor or major changes within the selected zone of document.

Keywords: Change Monitoring, Generalized Tree Comparison, Zone Selection of Web Page, Web Page Change Detection, Web Page Monitoring, Web Mining, Node Comparison, HTML, XML.

1. INTRODUCTION

Internet forms the important form of Information and Communication exchange. Everyone using internet is interested in a very specific type of data and it can be said that user visits some of these website regularly so as to track if some information has change overnight. If the web page changes too often, then it is cumbersome for the user to visit these sites frequently. For example, a user want to know about the latest recruitment in particular field related to particular location in the recent time. In this case, the user wants to be notified of the changes related to various recruitment details present on different web pages. In general, the ability to specify changes to this particular location in the document called here as zone of the document and notify it to user in different ways which will be useful for reducing the inefficient navigation. Due to rapid changes in the content of the web pages it has become very necessary to develop a Web Page Detection System which can detect these changes in the selected zone of the Web page in minimum browsing time.

There are basically four types of changes.

i. ***Structural Changes:*** It occurs when some HTML tags have been added or deleted in the web pages. Structural Elements like   <header> <meta> <article> and their attributes are used.
ii. ***Content Changes:*** It occurs when the content or information of the web page has been added, deleted or updated. Content Categories like Metadata, Flow, Sectioning, Heading, Phrasing etc. are used.
iii. ***Presentation Changes:*** It occurs when the design or appearances of the web page have been changed but the content or the information have not been changed. The tags of <b>,<i>,<p>,<h1>,<u>,<font>,<strike> etc are used.
iv. ***Behavioral Changes:*** Behavioral changes have been occurred when the active components such as applets, scripts, etc. have been changed [2].

A web page change detection system helps to reduce the browsing time of the user and allows the user to find the items in web page which change frequently. Change detection systems should provide the possibility of specifying the changes the user is interested in, to select the region of the document of interest, the items inside the region whose changes have to be monitored, and conditions on the type of changes which must be detected. Systems detecting changes on HTML pages with fixed structure are not able to satisfy these kinds of user needs since the page regions considered depend on the user's request. Current techniques for detecting document differences are computationally expensive and unable to focus on the portion of the page that is considered interesting for the user. Our technique represents the document as a tree and permits the user to focus on specific portions of it e.g. sub-trees. In particular the aim of our technique is that of efficiently monitor changes on small portion of a web page for instance of a job in recruitment process.

The main objective is to find the zone in the downloaded entire modified web page most similar to that selected by the user. This comparison process will be performed by generating the trees for selected zone of old web page and newly downloaded web page [4].

There are various types of properties which should be considered for detecting the changes in different versions of the web page like speed, accuracy, complexity, storage space, effective version management etc. [3]

2. Literature Survey

Number of research papers was found that handled the design of efficient algorithms for detecting changes in Web pages.

1) ***Document Tree based Approach*** [1] detects the structural and content changes. This Tree based approach is good for comparing the nodes of both the tree as old web page tree and modified web page tree. It gives the relevancy to the web pages and notifies the user about detecting the changes. For detecting structural change document tree is constructed and then signature values assigned to the root nodes and child nodes of the old and new web pages are compared. And for detecting content change first calculates the ASCII value of each character and then it is divided by those particular characters which are occurred in web page only once. Then it determines the text code for the different versions of the page and compares them. This algorithm defines the good comparison study for the different algorithms and provides simple method for detecting changes. Limitation of this paper is that comparison becomes longer if numbers of nodes are increased. So it's difficult to compare signature for each and every child node.
2) ***Optimized Hungarian algorithm*** [4] introduced three running time optimizations that control the operations of the Hungarian by considering time and accuracy analysis. This algorithm focuses on finding the most similar subtree, finding out of order tags or unclosed tags, edit scripting to find minimum edge weight monitoring for bipartite graph. Three measures for detecting changes are also considered which are *intersect*(percentage of similar words) ,*typedist* (position of elements),*attdist*(relative weight of similar attributes). This algorithm also defines that performance is inversely proportional to the depth of tree. Limitation of this algorithm is that running time may be large.
3) ***BIODIFF*** [8] algorithm covers some limitation of X-Diff algorithm. Unlike X-DIFF algorithm, BIODIFF designed for genomic and proteomic data. It outperforms app. 1.5 – 6 times faster than X-DIFF for different datasets. But one limitation of this algorithm is that If a database has more nodes that require min-cost max-flow matching, the improvement of Bio Diff is less as compared to X DiFF. Another limitation is that it takes more time to assign different matching types to the nodes in XML tree.
4) ***XML TREE DIFF*** [9] algorithm presents support for change control in the context of the Xyleme project that is investigating dynamic warehouses capable of storing massive volume of XML data. This algorithm is efficient in speed and memory space. It uses operations such as change node, delete node and insert node. Delta is constructed to find the matching of nodes between two trees. The use of XML specificities in algorithm leads to significant improvements. Drawback of this algorithm is that there is some loss of quality. Another drawback is that there is need of gathering more statistics about the size of deltas and in particular for real web data.
5) ***CH-DIFF*** **and** ***CX-DIFF*** [10] are developed by the Webvigil, a system that automates the change detection and timely notification of HTML/XML pages based on user specified changes of interest. *CH-DIFF* detects changes to various components such as links, images, keywords, phrases and any change using Longest Common Subsequence (LCS). But using LCS will be computationally expensive. *CX-DIFF* algorithm consists of steps like object extraction and signature computation, filtering of unique inserts/deletes and finding the common order subsequence between the leaf nodes of the given trees. Assorted and Linked Monitoring is also introduced in the paper. Immediate, Best-effort, Interval based, Interactive notifications are provided to the user. Drawback of this paper is that expensive computation and sentinels or user requests can be overloaded on the single server.
6) ***Level Order Traversal*** [11] is another form of the breadth first traversal. It includes document tree construction, document tree encoding and tree matching (based upon the concept of R.M.S. value of the content), for the detection of structural changes and content changes. Parameters used in this algorithm are node id, child node, parent node, level, tag name, content value or RMS value (sum of multiply the position of the character with its ASCII value). It has linear time complexity because it traverses only the changed portion of the tree rather than the whole tree and hence saves the time. It also extracts effectively the changed content from different versions of a web page. It is simple, less cost, and understandable and can reduce the network traffic by using HTTP metadata. It can successfully retrieve the summary but not the complete content of the newly created page which is insufficient information for the user.
7) ***Hashing*** **based** [12] saves computation time by limiting the similarity computations between two versions of a web page to nodes having the same HTML tag type, and by hashing the web page in order to provide direct access to node information. To speed up the process of web change detection system a hashing based technique is used for direct lookup of subtree node information during comparisons, and eliminated irrelevant node comparisons by limiting them to nodes of the same type. This algorithm is also applied to RSS (really simple syndication) feeds change detection for delivering regularly changing web content, such as news. Original and enhanced approaches are introduced to improve the comparisons. This algorithm suffers from one limitation that inability to detect changes when root tag is changed. Multithreading is used for improving the performance. This algorithm can be further implemented on dynamic web pages and can also be defined in XML file as future work.
8) ***Tree traversing*** [13] is developed for detecting the structural as well as content change. This algorithm is divided into two parts tree development and change detection. The proposed algorithm used bottom up approach for assigning hash value to each leaf node and tag value to the non-leaf nodes. This algorithm include extracting the tags from html code of the page, assigning node number to each node, finding multiple child of node, calculating the hash value and tag value by assigning hash function, comparing tags. This algorithm is based on depth first search. This algorithm is simple to understand and it saves the browsing time. But the drawback of this algorithm is that performance is not defined when depth or levels of the tree is increased.
9) An existing product is ***Copernic Tracker*** [5] that seems to be the most mature software aimed at monitoring Web

sites. The software can track changes in the text and images and monitors for the presence of specific text. The system however does not allow for specifying how much emphasis to place on monitoring different aspects of the Web page and does not provide a utility for monitoring a specific region of the web page. This product does not reveal performance data that discusses speed or accuracy.

10) A ***Website-Watcher*** [6] which includes the ability to monitor password protected web pages. The system offers limited freedom for selecting a zone to monitor and lacks a proper user interface to show the changes. This system also does not provide objective performance data other than subjective user reviews.
11) ***WYSIGOT*** [7] which is a commercial application that detects changes between HTML pages. This system has to be installed on the local machine and the granularity of change detection is at page level.

3. General Design

3.1 Overview

Web page change detection system for selected zone based on tree comparison mechanism corresponding to HTML pages. It uses generalized tree Comparison Algorithm to compare the trees. Algorithm uses the properties of HTML page and heuristics as a node of the trees. This proposed system includes functionalities and interfaces for processing user request, fetching web page from the internet and allowing users to select zone in a web page to monitor. In addition to above this proposal highlights changes on the web page of selected zone being monitored.

3.2 General Architecture

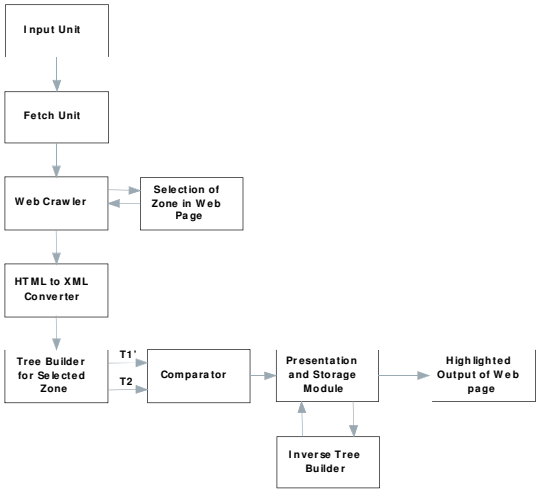


Figure 1. Architecture of Web page Change Detection System for selected Zone.

The figure 1 depicts Architecture of the working model of Web Page Change Detection System.

The step by step working model of architecture is represented as follows:

Step 1. Fetching of old Web Page and Generation of Tree

The web crawler will fetch the old HTML web page which user wants to monitor and the tree will be generated for complete web page.

Step 2. Selection of Zone of old Web Page [14][15].

The system provides the zone selector tool to allow user to select zone to monitor. For the selected zone the user will specify which portion of web page to monitor and the period of time between two successive monitoring tasks.

Step 3. Generation of Tree for the selected zone of Webpage

We have develop tree builder which will be responsible for converting of web page to tree data structure in which node contain tag and their attribute, while leaves contain their text content. The process of building the tree is described in section 3.3.1. The data will be generated from the tree generated for old HTML web page. The tree for the selected zone of the Web Page is also stored.

Step 4. Fetching of Modified web page.

After completion of time slot which is set by user, the browser will download the web page which is to be monitored.

Step 5. Generation of modified tree for fetched web page.

The system will generate the tree for complete modified web page using Tree Builder.

Step 6. Checking the sub tree for change in content attribute & layout.

Let T1' be a abstract general tree generated by user corresponding to the zone selected by user and T2 tree will be for the modified web page. In the comparator module, comparator will find the most similar zone in the new web page. In this comparison T1' the subtree of selected zone of Old Web page is searched in T2 Tree of Modified Web Page and it outputs corresponding changes in subtree of T2.

In Comparator module we are using the following modules

i) **Text Compare Module** for comparing content i.e. HTML text.

ii) **Attribute Compare Module** that will compare HTML attributes.

Step 7. Change detection and report generation:-

Presentation and Storage module stores the result & reports after every monitoring period using the Inverse Tree Builder. Output Center will notify user about the detected changes.

3.3 Modules

3.3.1 Tree Builder

We have developed tree builder which will be responsible for converting of web page to tree data structure in which node contain tag and their attribute, while leaves contain their text content. The process of building the tree is as shown in figure 2.

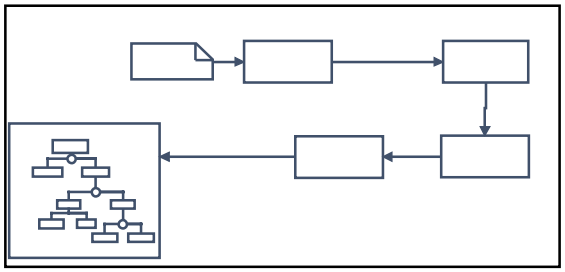


Figure 2. The Tree Builder Sub-Module

This will consist of four modules:-

i) Filter
ii) Re-Arranger
iii) HTML to XML Converter
iv) Tree Builder

i) **Filter:-**It filters the HTML document from all irrelevant content like comments and non HTML content e.g. Script, applet etc.

ii) **Re-Arranger:-**It will re-arrange HTML document to fix out of order tags e.g. <a>text1text2text3text4 .The tags <a> and are terminated out of order which results in an improper tree representation of this code.

iii) **HTML to XML converter:** It will convert the html webpage to XML webpage.

iv) **Tree Builder:-**This step will generate trees from the processed HTML pages.

The example of the Tree builder can be explained as follows

```
<html>
<head>
<title>Dept. of Tech</title>
</head>
<! This is comment>
<body bgcolor=skyblue>
<h1> Dept. of Tech </h1>
<h2>Shivaji University </h2>
</body>
</html>
```

Original HTML Page

```
- <html>
- <head>
  <title>Dept. of Tech</title>
  </head>
- <body bgcolor="skyblue">
  <h1>Dept. of Tech</h1>
  <h2>Shivaji University</h2>
  </body>
  </html>
```

Converted XML Page

The Figure 3 represents the structure of the document we just created. This document is based on document object model which is describes in 3.3.2. An XML document has a single root element which contains the document's entire element.

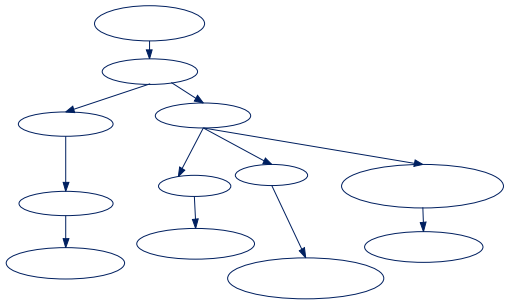


Figure 3. Tree structure of an XML document

In Figure 3 three kinds of nodes are observed in the DOM tree. They are element, text, and attribute nodes. Text nodes are leaf nodes with one value. Attribute nodes are non-leaf nodes, but they have two labels name and value. According to the DOM specification, element nodes and text nodes are ordered while attribute nodes are unordered.

3.3.3.2 Pseudo code of Tree generation

The Tree Generation Pseudo code is as follows

Tree generation Pseudo code

1. Read Xml document into XmlNode

2. Generate the modifiedTreeNode

3. Add XmlNode to the collection of tree Nodes

4. Check the type of XmlNode

4.1. Processing instruction or Xml declaration Set modifiedTreeNode Text="<?" +

XmlNode Name and XmlNode Text

4.2. IF element set modifiedTreeNode Text = "<" + xmlNode Name +">"

4.3. Attribute Set modifiedTreeNode Text = "ATTRIBUTE: " + xmlNode Name

4.4. Text and CDATA Set modifiedTreeN ode Text = xmlNode Value

4.5. Comment Set modifiedTreeNode Text ="<!-"+xmlNode Value+"— >"

5. Read attribute collection

6. For each attribute Go to step 1

7. Read ChildNodes Collection

8. For each ChildNodes Go to step 1

3.3.2 *Document Object Model*

Definition 1 (Document tree): A document tree is a tuple T=<N, p, r, l, t, a> where

(1) N is the set of nodes of the tree,
(2) p is the parent function associating each node (except the root r) of the tree with its parent,
(3) r is the distinguished root of T ,
(4) l is a labeling function from leaf (T) to $\sum^{+}$
(5) t is a typing function from N to τ and
(6) a is an attribute function from N to A* $\sum^{*}$

Thus, a document tree is an unordered tree whose nodes (elements) are characterized by their markup type and the

associated set of attribute-value pairs. Leaf nodes have associated the actual textual content of the document. Given a document tree T , whose root is r, and a node en of T, we denote with T(en) the sub-tree of T rooted at e_n. However, to be suitable for a change detection tools, this model should be extended by associating more information to internal nodes of the tree. We define two functions characterizing an element w.r.t. the whole document tree, type (e_n) and w(e_n). If r, e_2, . . . , e_n is the path from the root r to the element e_n, type (e_n) =t(r) t(e_2) …. t(e_n) whereas w(e_n)={s|s is a word(substring separated by blank to the other substring) contained in l(e) ∧e ∈ leaf (T (e_n))}. We also define a (e_n) as the set of attributes associated to en. Essentially w (e_n) is a set of words contained in the various text strings associated to the leaves of the sub-tree rooted at en, and type(n) is the concatenation of type label in the path starting from the root of the tree and ending in en, i.e. the complete type of the element [14].

3.3.3 Zone Selection Tool

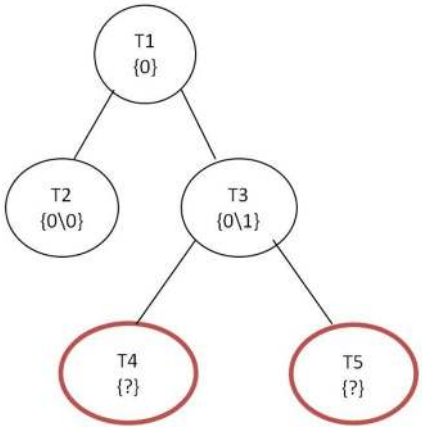
User first input the Web Page and then selects the zone of his interest. After selection of zone, the Tree is generated for Web Page and positions are assigned to all the nodes of entire tree starting from root node of tree. Positions are nothing but it is relationship between parent and child element of tree. The positions are assigned using AssignPosition algorithm. In addition to that, tree for selected zone i.e. Tz is prepared. Now text of Root of Tz is compared in the tree of Old Web Page. When search is found the position of that node i.e. selected zone's node in the Old Tree T1, it's position is recorded in the position variable Pos.

3.3.3.1 Pseudo code of Zone Selection

Positioning (AssignPosition)

1. If tree node T is root node, then assign index number to tag of T
2. Else, concatenate tag of parent node P and the index number of T
3. Assign the concatenation to the tag of T
4. Recursively call AssignPosition on every child node

The steps of execution of assign position is described using Figure 4



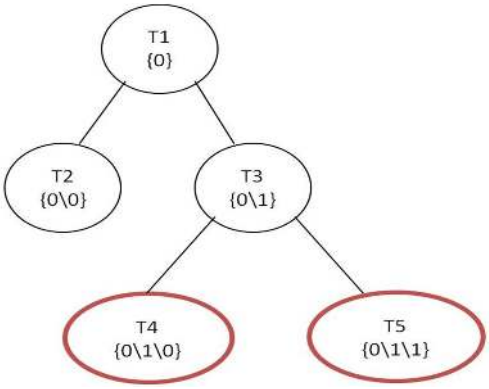


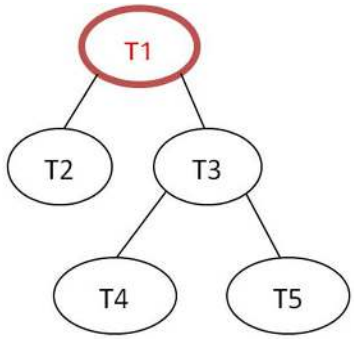
Figure 4. Typical Tree for Assigning Position.

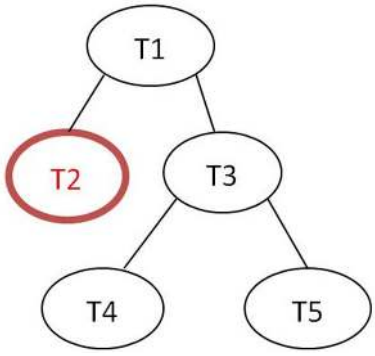
In Step1, first node traversed is the node T1. It has no parent node. Hence, the root node T1 is assigned position **{0}**. In Step2, the next node traversed is the node T2. It has parent T1. Hence it is assigned the position **{0\0}**, which indicates, the 1st child node under the parent node. In Step3,the node T3 is similarly assigned the value **{0\1} .** In Step4, the node T4 is assigned the position **{0\1\0}** and T5 is assigned **{0\1\1}.**

3.3.3.2 Pseudo code of retrieving the position from text (GetPosByText)

Retrieving the position from text (GetPosByText)

1. Compare given text with text of treenode T1
2. If text matches, then return the position from tag property
3. Else, recursively call GetPosByText on each child node





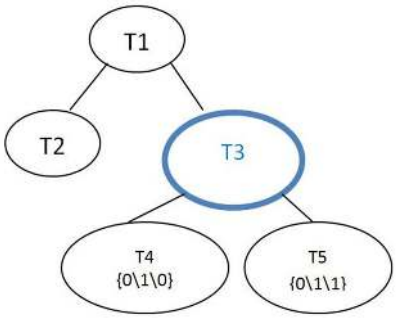


Figure 5. Typical tree for GetPosByText

Steps of execution of GetPosByText are described using Figure 5.

The node with text "T3" is to be searched. In Step1, the root node T1 is compared with "T3". It does not match the given text. Hence it is skipped.

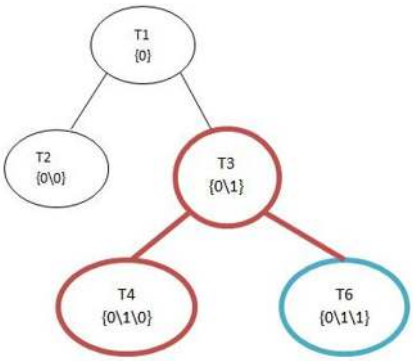
In Step2, the next node T2 is then compared with the same procedure. It also does not match the given text "T3". Hence this node is also bypassed.

In Step3, the next node T3 is then compared. This node matches the given text "T3". This ends the search and node T3 is returned.

3.3.3.3 Pseudo code for Zone selection

1. Prepare a sub tree for selected zone from the root node T of selected zone T1(Tz)
2. Retrieve the position of the selected zone's root node i.e T1(Tz) in the tree of Old Web Page
3. Save the position in a temporary variable Pos
4. Assign positions to the tree of Modified Web Page using AssignPosition algorithm.
5. Traverse T1 from root and find T1(Tz) using position "pos" from the tree of Old Web Page
6. Traverse T2 from root and find T2(Tz) using position "pos" from the tree of Modified Web Page
7. Compare the tree nodes T1(Tz) and T2(Tz) using Generalize tree comparison.

Here T1(Tz) and T2(Tz) are treated as root nodes and generalized comparison starts comparison and find the changes in Modified Web Page for selected zone i.e. T1(Tz). The steps of execution of zone selection are described using Figure 6.



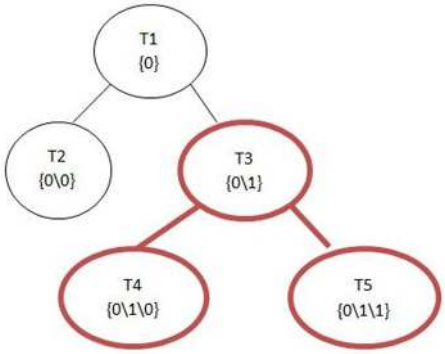


Figure 6. Tree For Zone Selection.

In Step1, the user selects the zone. Then the main tree (with the positioning) is selected and the position of the root node of selected zone is recorded. The figure 6 shows the selected zone containing the nodes T3, T4, T5. Here the root node of selected zone is node T3. Hence the position **{0\1}** is recorded.

In Step2, the Next, the node at the recorded position i.e. at the position {0\1} is searched in the tree of Modified web page.

In Step3, using the tree comparison both the sub trees are compared. T3 matches with node T3, T4 matches with node T4.

In Step4 whenever a mismatch is found, the change is recorded. The node **T5** does not match with the node at same position i.e. the node **T6**. This change is then recorded.

The stepwise execution of change detection of selected zone is described using typical trees of Old Web page and Modified Web page as shown in figure 7, 8,9,10.

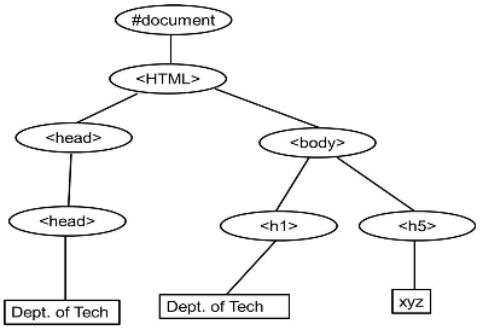


Figure 7. Typical tree of Old Web Page.

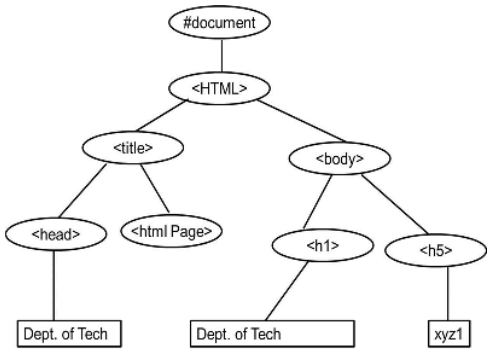


Figure 8. Typical tree of Modified Web page

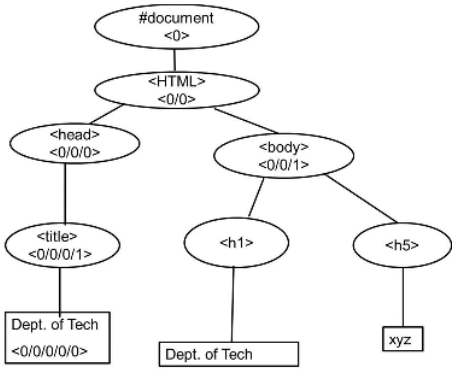


Figure 9. Assigned positions to the tree

For inputted Old Web page, users selects zone and its Sub tree is shown in Figure 10

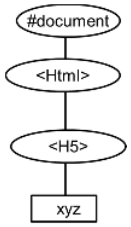


Figure 10. Tree for Selected Zone

Now T1(Tz) is find i.e (h5) in T1 using GetPostBy(Text). It selects text "<h5>" and searches this text in tree of Old Web Page. Searching starts from the root node i.e. "#document" then "#<html>." …….so on the from left to right. It locate the position of text "<h5>" in T1 tree i.e. position of <h5> i.e. {0/0/1/2} . Now it Save the position of <h5>i.e. {<0\0\1\2>} in variable Pos. Now Modified Page is inputted and it's tree is given in Figure 8 Positions are assigned to tree T1 are shown in Figure 9. Using variable Pos, search selected zone's node in T2 tree i.e. T2(Tz)=Pos. Now, applying Generalized tree comparison algorithm By using T1(Tz)-- <h5>----<xyz> and T2(Tz)----- <h5> ----- <xyz1>, Change detected in xyz and xyz1.

3.3.4 Comparator

The Comparator has the task of performing the bulk of the change detection process. To perform this, an algorithm is developed for generalized tree comparison algorithm. This algorithm uses trees generated from Tree Builder of HTML pages and detects the changes automatically.

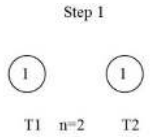
For the generalized tree comparison algorithm, the inputs are T1' and T2 two abstract general trees generated by the Tree Builder module discussed earlier. They correspond to the selected zone of Old Web Page and to the Modified Web Page. Now the Comparator finds the most similar sub tree as that of selected zone in the modified web page .Then these two trees are compared. When comparison is finished, it outputs changes in T2 tree as content change, attribute change.

Our Comparator matches each node of selected zone in the Old version i.e. T1' with its corresponding MSST node in the Modified version i.e.T2. Then the entire search space will be explored to detect changes incurring high execution time. This is a tree to tree correction algorithm by defining the ordered mapping between trees. This algorithm gives the best possible matching between two trees but with a run time of $O(n^2 h^4)$ where $n= \max(n_1,n_2)$ $h= \max (h_1,h_2)$, h_1=height of T1 and h_2= height of T2 [20].

3.3.4.1 Pseudo code of Tree Comparison

1. Load T1' and T2
2. Count the nodes of selected zone of T1' and MSST of T2
3. Assign lowest value to n
4. Traverse T1 and T2 up to n
5. Read the text for current node of T1 and T2
6. Compare both text
7. If text are note equal than current node has difference
8. And if text are equal call same algorithm recursively for child

The steps of execution of Pseudo code of Tree Comparison is given in Figure 11.



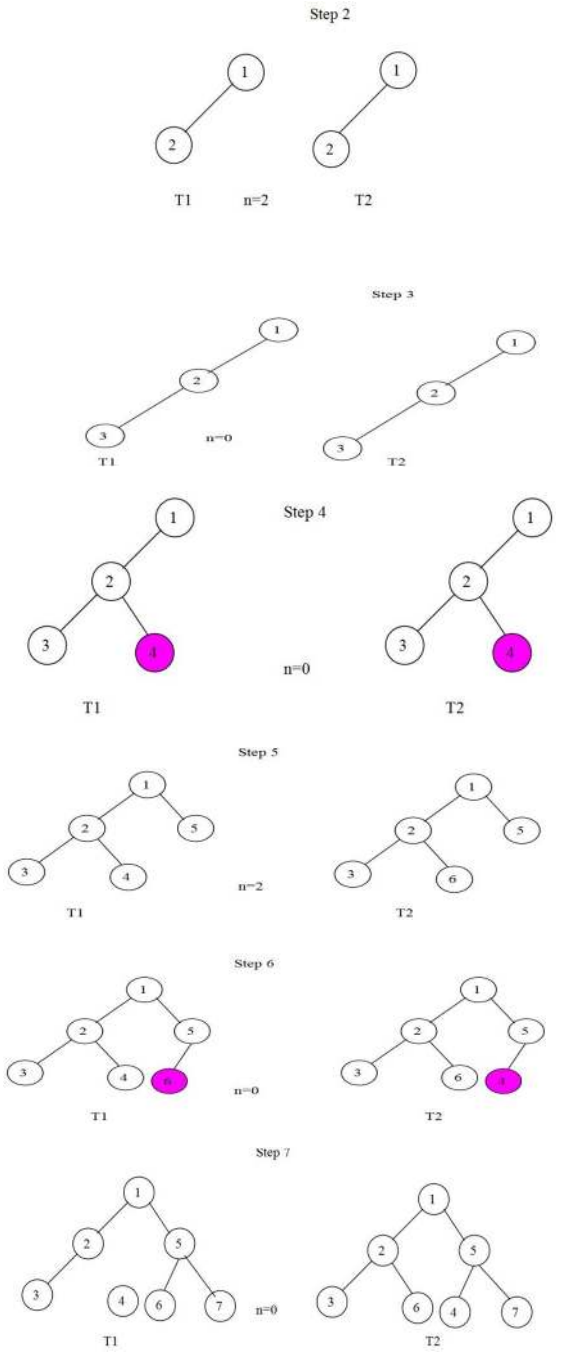


Figure 11. Steps of Execution of Pseudo of Tree comparison execution.

4. Implementation

The Application which is discussed in this paper is developed using the C# language and .NET platform. The Typical web Page considered for an experiment is as shown in figure 12. The old Web page and the modified Webpage is given as input to the system.

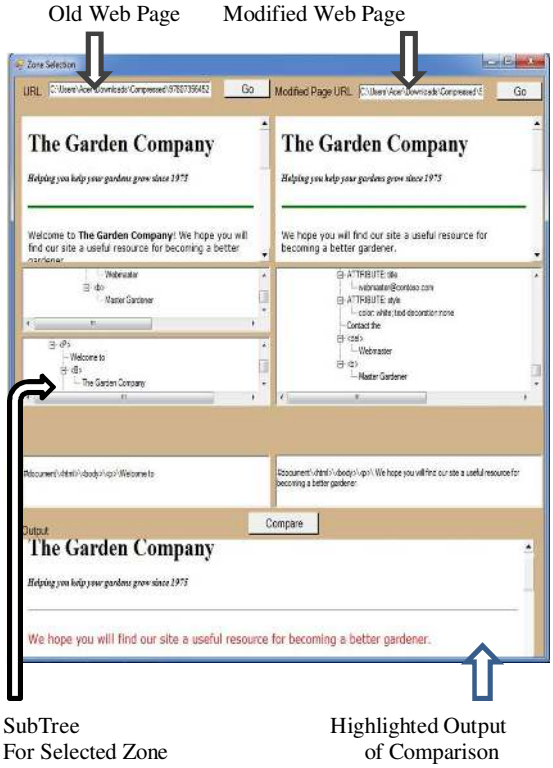


Figure 12. Web page Change detection for Selected Zone

5. Conclusion

The Web Page Detection system is for selection zone using generalized Technique detects changes for selected zone. It detects changes for text and attributes at minute level. We have developed generalized tree comparison Algorithm for selected zone. It helps us to reduce browsing time. It shows the output of change text in red color. This saves the time of detecting the changes in the entire Web page. Such system is useful as described above for stock broker, job seekers who are continuously monitoring the changes in the web pages.

6. References

[1] Naveen Kumar Varshney, Dilip Kumar Sharma "A Novel Architecture and Algorithm for Web Page Change Detection.

[2] Naveen Kumar Varshney, Dilip Kumar Sharma "An Enhanced Architecture and Algorithm for Web Page Change Detection"

[3] Shobhna, Manoj Chaudhary "A Survey on Web Page Change Detection System Using Different Approaches" International Journal of Computer Science and Mobile Computing*", Vol. 2, Issue. 6, June 2013, pg.294 – 299.*

[4] Imad Khoury, *Student Member*, *IEEE*, Rami El-Mawas, *Student Member*, *IEEE*, Oussama El-Rawas, Elias Mounayar, and Hassan Artail1, *Member*, *IEEE* "An Efficient Web Page Change

Detection System Based On An Optimized Hungarian Algorithm".

[5] Copernic Technologies, Copernic Tracker Product, 2006, http://www.copernic.com/en/products/tracker/tracker-features.html.

[6] M Aignesberger WebSite-Watcher Product, http://www.aignes.com, 2006.

[7] Wysigot, http://www.wysigot.com.

[8] Song, Bhowmick. "BioDIFF An Effective Fast Change Detection for Genomic and Proteomic Data" in a Proceedings of the Thirteenth ACM conference on Information and knowledge management., Pp146- 147, Nov. 2004.

[9] G. Cobena, S. Abiteboul, and A. Marian, "Detecting Changes in XML Documents," Proc. 18th Int'l Conf. Data Eng., pp. 41-52, 2002.

[10] S. Chakravarthy, Subramanian, "Automating change detection and notification of web pages" , Proc 17th Int'l Conf. on DEXA, IEEE , 2006.

[11] D.Yadav,A.K. Sharma, J.P. Gupta"Change Detection In Web page" in a proceeding of 10th international conference on information technology,pp 265-270,2007.

[12] G. Srishti, Rinkle R. A. "An efficient for web page change detection" , IJCA, VOL. 48, NO.10, June.

[13] Marouane Hachicha and Jerome Darmount, Member IEEE Society "A Survey of XML Tree Patterns" IEEE Transaction on

[14] S. Flesca, E. Masciari "Efficient and effective Web change detection" Data & Knowledge Engineering 46 (2003) 203–224.

[15] Raihan Al-Ekram, Archana Adma, Olga Baysal "diffX: An Algorithm to Detect Changes in Multi-Version XML Documents".

[16] Latifur Khan, Lei Wang and Yan Rao "Change Detection of XML Documents Using Signatures" .

[17] Documents Gr´egory Cob´ena, Serge Abiteboul Am´elie Marian"Detecting Changes in XML".

[18] Monika Yadav Mr. Pradeep Mittal "Web Mining: An Introduction" Volume 3, Issue 3, March 2013 ISSN: 2277 128X International Journal of Advanced Research in Computer Science and Software Engineering.

[19] *M.Srividya, D.Anandhi M.S.Irfan Ahmed* "Web Mining and Its Categories – A Survey" *International Journal Of Engineering And Computer Science ISSN:2319-7242 Volume 2 Issue 4 April, 2013 Page No. 1338-1345* .

[20] Geeta R. Bharamagoudar, Shashikumar G.Totad, Prasad Reddy PVGD "Literature Survey on Web Mining" *IOSR Journal of Computer Engineering (IOSRJCE) ISSN: 2278-0661, ISBN: 2278-8727 Volume 5, Issue 4 (Sep-Oct. 2012), PP 31-36.*

[21] Mr Dushyant Rathod "A Review On Web Mining" IJERT ISSN:22780181 Vol.1 Issue 2 April 2012.

[22] Manoj Pandia, Subhendu Kumar Pani, Sanjay Kumar Padhi,Lingaraj Panigrahy, R.Ramakrishna "A Review Of Trends In Research On Web Mining" *International Journal of Instrumentation, Control & Automation (IJICA), Volume 1, Issue 1, 2011*

[23] Adam Rae, Vanessa Murdock, Adrian Popescu, Hugues Bouchard "Mining the Web for Points of Interest".

[24] Chapter 5 "Proposed Method for Web Page Change Detection".

[25] Web Mining— Concepts, Applications, and Research Directions *Jaideep Srivastava, Prasanna Desikan, Vipin Kumar Chapter 21.*