An Efficient Algorithm for Web Page Change Detection

Srishti Goel
Department of Computer Sc. & Engg.
Thapar University, Patiala (INDIA)

Rinkle Rani Aggarwal
Department of Computer Sc. & Engg.
Thapar University, Patiala (INDIA)

ABSTRACT
Internet is actively used for the exchange of information. People upload the web pages and updating the new web pages very frequently. There is a frequent change in the content of the web page hence it become necessary to develop an efficient system which could detect these changes efficiently and in the minimum browsing time. So as to achieve this we compare the old web page and the new web page. Changes in a web page can be detected with the use of various algorithms. Various tools and services are also available which can be used to detect these changes.

In this paper a new algorithm for the structural as well as content change detection has been proposed and described. For better results tree has been designed for the corresponding web pages. The proposed change detection algorithm is based on assigning hash value to each leaf node and tag value to the non leaf nodes. Bottom up approach has been used for assignment. The level of each node has been used to find hash values and modification in a node. It has been shown with the help of suitable examples that the proposed algorithm extracts the changes very efficiently from the various web pages.

Keywords
Web page change detection, Tag of node, Tree matching, Hash value, Change monitoring

1. INTRODUCTION
Internet has become an effective way for the exchange of information across the world. It delivers the information mainly in the form of the web pages. New pages are uploaded frequently to provide new and more information to the user. Now the users are not only interested in gathering the new information along with the changes that have taken place in the web page. If a user is observing a web page that is refreshed very frequently then the user might not be interested in downloading or viewing the entire web page each and every time. He will be interested in observing the changes that have taken place in the web page since last visit. In order to find these changes user will select a zone in which he is interested to observe the changes. The web page detection system will find the zones selected by the user in old and new web pages. The system then compares the two web pages to detect the changes which have taken place.

User may be interested to know the changes in the web pages every time he visits the page. For example: a stock trader may be interested in knowing the current status of the market or he may be interested in knowing the new price of the stocks. Similarly a user may be interested in knowing the ranking of the cricket teams, statistics of players playing in IPl which changes frequently with each match. The web page change detection system helps the user to detect such changes efficiently and in minimum browsing time.

Various algorithms [1][2][3] and tools [4][5][6] are available to detect the changes in web pages. Different algorithms are proposed in various research papers. Erwin Leonardi [7] explains the change detection based on ordered and unordered tree. In this paper a scheme is proposed which help to detect the changes. It also explains the types of changes which can occur in a web page and the architecture which can be used to detect the changes. The hash based algorithm is to be used to detect the changes. First a web page is searched in which changes had to be detected. Then the tree will be designed for that web page and then the two trees are compared by the tag values assigned to each node.

2. CLASSIFICATION OF CHANGES IN A WEB PAGE
Web pages can be classified as follows:
1. Structural Changes
2. Content Or Semantic Changes
3. Presentation Or Cosmetic changes
4. Behavioural Changes

2.1 Structural Changes
These changes occur when ever we a tag is added or deleted in a web page i.e. addition or deletion of a tag causes structural change[8] in a web page. Sometimes the addition/deletion/modification of a link also causes a structural change. These types of changes are important to find as they are not visually perceptible.

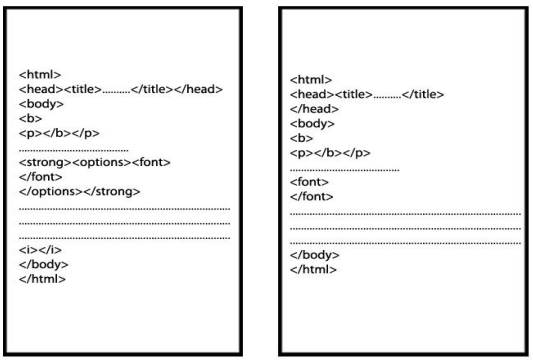


Fig 1: a) Initial Version b) Modified Version

2.2 Content or Semantic Changes
These changes occur whenever the content of a web page changes according to the user point of view. [8] A stock trader may be interested to know the changed status of the market or the current price of the share. He is interested in the current or

the changed status of the market and not in the old price or the old market status. Another example could be the web pages displaying the score of a match. The user viewing the page is interested in the current score and the content of the page changes every time whenever the score changes. Web pages containing the records or schedule of a tournament changes accordingly with the change in a tournament.

```
<html>
<head>
<title>PORTFOLIO</title>
<Link href = "style.css" rel = "stylesheet" type = "text/css" /></head>
<body>
<P>Welcome To My World</P>
<P> Hiiiiiii to all..... </br>
I am a first year B Tech student studying in the LNMIIT </p>
  <p>( LNM Institute of Information Technology, Jaipur).<br />
   Pursuing Electronics and Communication Engineering course.<br />
   The 'My Bio' page describes myself, my thoughts, interests<br />
   and links to connect with me socially. </p></body>
</html>
```

(a)

```
<html>
<head>
 <title>PORTFOLIO</title>
<link href="style.css" rel="stylesheet" type="text/css" /></head>
<body>
<p> Welcome to cricket World.</p>
 <p>Today's score<br /></p>
 <p>INDIA V/S ENGLAND<br />
  India played very well and beat England by 6 wickets<br />
   England made 253 runs in 46 overs and India made 254 runs nad beat
England by 6 wickets. </p></body>
</html>
```

(b)

Fig 2: a) Initial Version b) Modified Version

2.3 Presentation or Cosmetic Change

These Types of changes Occur whenever the appearance of a web page changes but the content of a web page remains the same [8]. For example: with the changes in tags the appearance of a page may change without change in the content of a page.

```
<html>
<head><title>.....</title>
</head><body>.....</body>
<p>my first web page</p>
<p>Page change detection system
</p>
</body>
</html>
```

(a)

```
<html>
<head><title>.....</title>
</head><body>.....</body>
<p style ="backgroundcolor = red"><u>my first web page</u></p>
<p>Page change detection system
</p>
</body>
</html>
```

(b)

Fig 3: a) Initial Version b) Modified Version

2.4 Behavioural Changes

Behavioural changes refer to changes in the active components which are present in a document[8]. For example, web pages may contain scripts, applets etc as active components. When such hidden components change, the behaviour of the document gets changed. However, it is difficult to catch such changes especially when the codes of these active components are hidden in other files.

3. ARCHITECTURE

The general architecture is shown below in the figure 4. This architecture contains the comparison module where various algorithms can be applied to compare the two web pages. [9] This architecture follow a path from the start state to the end state these states are denoted by the ellipse. The user will input the web pages in which the changes is to be detected. The user will input the old web page and the new web page. The input module then asks the crawler to fetch the pages. The old web page will be fetched from archive and the new web page will be fetched from the site. These pages will be saved in page report archive. These pages will be sending to the tree builder module via manager. Tree builder module will design the trees corresponding to the old and the new web page. These trees will be passing on to the comparator module. In this module comparison algorithm is applied to find the difference between the two web pages. Then the result will be given to the presentation module. Presentation module prepares a report from these results and save it in page report archive. Notification centre will compare all the reports related to a page compile them and send it to the user. There is an inverse tree builder module which will generate the page from the tree and give the pages to the browser to show the result.

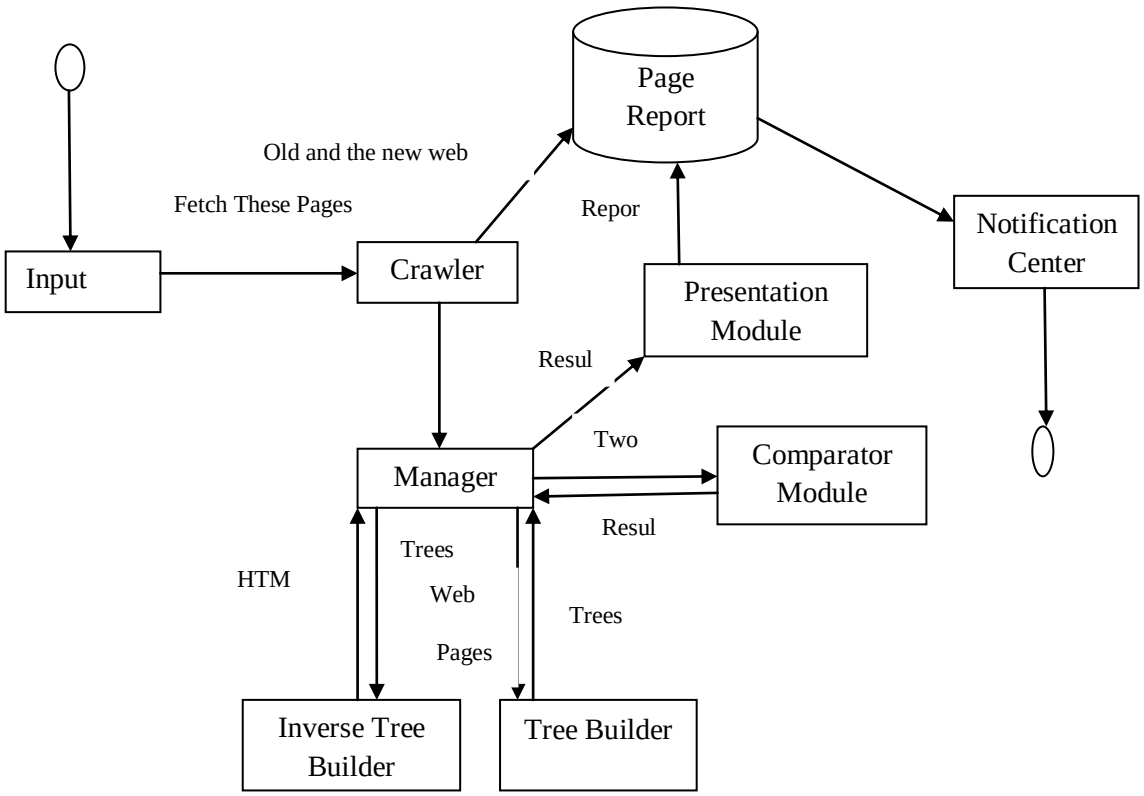


Fig 4: Architecture of a web page change detection system

4. PROPOSED ALGORITHM

An algorithm is designed which will find structural as well as the content change. Before finding the changes URL of a particular web page is to be searched. A crawler has been designed which will save the HTML code of the page. Another algorithm is designed which will develop the trees corresponding to the pages. Algorithm is mainly implemented in C# and uses the inbuilt classes and methods of C#.

Tree development algorithm is mainly based on tags extraction and then developing the tree while the change detection algorithm is based on assigning the hash value to each node which we have calculated by giving each tag a number and the level number where the child is zplaced in the tree. Hash value is mainly assigned to the leaf nodes in the tree and for the non leaf nodes the tag value will be assigned which is the sum of hash value of its child. Tag value is assigned in a bottom up manner and uses them to find the changes. If the tag value of a node changes then we can say that there is a change.

4.1 Algorithm for tree development

4.1.1 Extract Tags

Input: Web page searched by the crawler
Output: Tags are extracted from the web page

1. Find index No. of each tag
2. If tag does not exist then it will return -1.
3. Assign node No. And nodename to each node and then add the node to the tree
   node.index = index;
   node.nodename = "html";
   tree.Add(node);

4.1.2 Assign Node Number to each node

Input: Extracted tags are given
Output: Node Number is assigned to the tag

1. Check for each tag t
2. If
   It is a closing tag then we do not assign any Node number.
3. Else
   Count opening tag and closing tag until we get the closing tag of t
   if (countopen - countclose == 1)
   tree[j].nodeno = k; k++;

4.1.3 Find Childs

Input: Nodes will be given as a input
Output : Find the child of each node

1. Check for each tag t
2. If
   It is a closing tag then we don't assign nay child.
3. Else
   if (countopen - countclose == 1 &&
   !subs.Equals("/"))
   tree[i].childs = tree[i].childs + tree[j].nodeno + ",";

4.2 Algorithm to detect the change

4.2.1 To Calculate Hash Value

Input: Nodename is name of leaf node to which hash value has to assign, level is depth at which that node exists.
Output: Hash value of node.

1. Provide some number to all tags i.e. nodename which can exist in webpage or can take enumeration of tags.
2. Foreach node n in tree
   if (n.childs.Equals(""))
   n.tag = GetHash(n.nodename, n.level);

4.2.2 Assign Hash Function

Input: All nodes of a tree.
Output: Hash value of leaf nodes.

1. For each node of a tree.
2. If (node.childs.equals (null)).
   Call to calculate hash value function.
   Hash function = nodename + Levelno.

4.2.3 Calculate Tag

Input: All nodes of a tree.
Output: Tree with calculated Tag value of all nodes.

1. Start from bottom of a tree for each node n.
2. We will assign tag values to non leaf nodes.
   If (! node.childs.equals (null))
   n.tag = sum of tag values of its all child's

4.2.4 Find changes

Input: Old tree and new tree to find changes in them.
Output: Nodes added or deleted in old tree.

1. Find nodes at each level of tree in old tree as well as new tree.
   if (!n1.childs.Equals(""))
   nextchildnodes = nextchildnodes + n1.childs + ",";
2. Compare nodes at each level
   If number of nodes at any level of new tree are less than or equal to node at same level of old tree
3. Compare the tags of parents of that level
4. If tag of any new parent(n) > old parent(o)
   if (oldparentChild.Count() > newparentChild.Count())
   int d = n1.tag - n3.tag;
5. Then addednode=n.tag – o. tag
6. Else
   int d = n3.tag - n1.tag;
7. Deletednode = o.tag – n.tag.

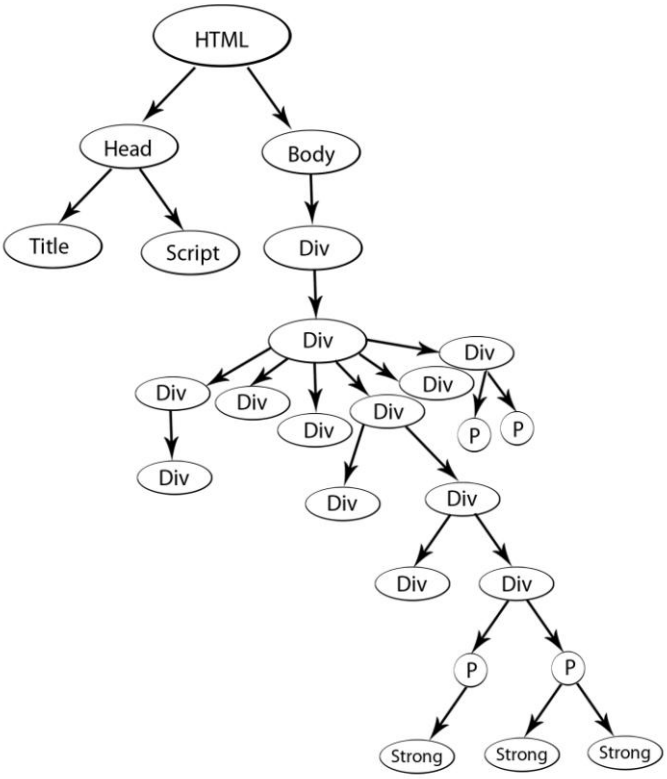


Fig 5: Initial Version

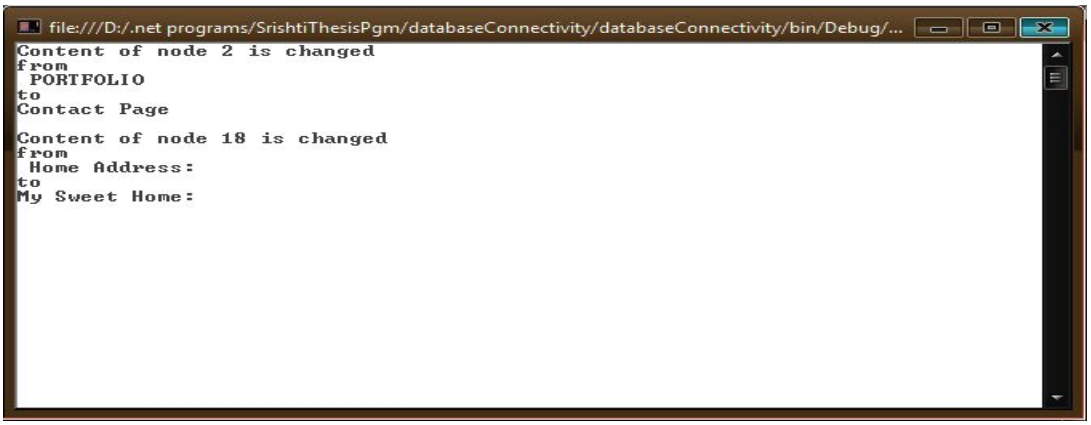
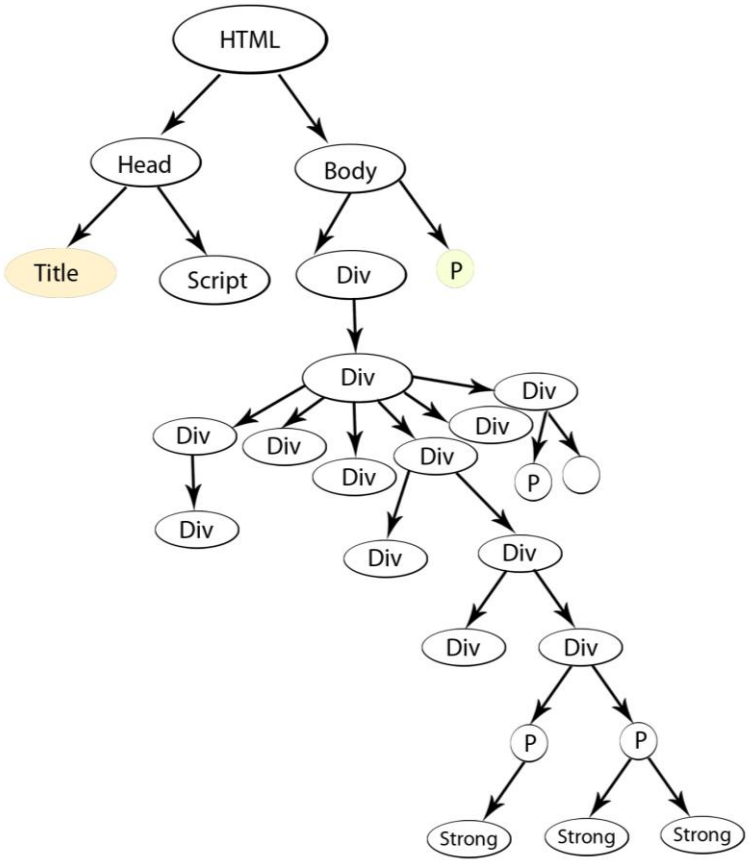
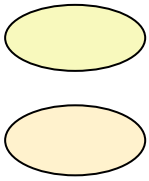


Fig 6: Snapshot of content change





New Node Added

Node Deleted

Fig 7: Modified Tree

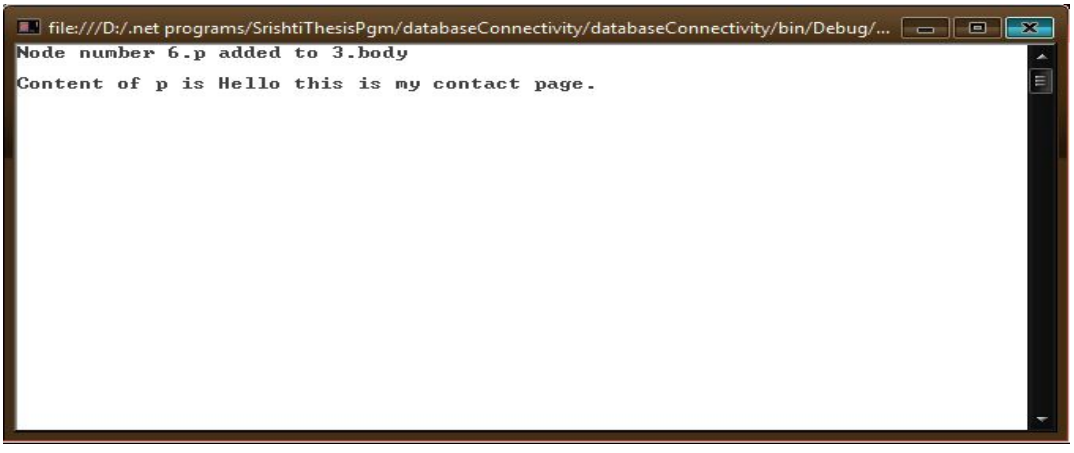


Fig 8: Snapshot showing structural change

Fig 9: Snapshot showing deletion of a node

5. CONCLUSION

This study is to develop an efficient web page change detection system which will detect the changes in a web page. The proposed algorithm extracts changes between different versions of web pages. The algorithm is able to make out structural as well as content based changes by developing an efficient application in C# which scores on other techniques on the basis of simplicity and understandability. This algorithm detects the changes, based on the change in the tag value. Tag value is assign to every node. To find the changes algorithm traverses each node in two version of the tree.

Tree traversing is based on the depth first search. Two different algorithms are used first one is used to construct the tree corresponding to the web pages and the second one is used to detect the changes which are made in the web page. While detecting the changes in the new and old web page three main problems are encountered. Solution for these problems has been provided through these algorithms. This system is useful for saving the browsing time and also gives the user time to time update regarding the changes which will occur in a web page.

6. REFRENCES

[1] Buneman P., Davidson S., Fan W., Hara C., and Tan W. Aug 2002"Keys for XML" Proceedings of international conference on Computer Networks, 39 (5), 473–487.

[2] Chawathe S., Rajaraman A., Garcia-Molina H. and Widom June 1996, "Change Detection in Hierarchically Structured Information", Proceedings of the ACM SIGMOD International Conference on Management of Data, 25(2), 493-504.

[3] Chawathe S., Garcia-Molina H. May 1997"Meaningful Change detection in structured data", proceeding in ACM SIGMOD International conference, 26-37.

[4] Liu, L., Pu C., and Tang W. "*WebCQ: 2000* Detecting and Delivering Information Changes on the *Web"*. In Proceedings of International Conference on Information and Knowledge Management, pp.512-519.

[5] Lu, B., Hui C.B., and Zhang Y.2002. Personalized "Information Monitoring over the Web". in First International Conference on Information Technology and Applications(ICITA).

[6] Available at: Mind-it, http://www.netmind.com.

[7] Leonardi E., Sourav Bhownick S., "Detecting Content Changes on Ordered XML Documents Using Relational Databases".

[8] Yadav D. 2009"Design of A Novel Incremental Parallel web crawler" Phd thesis, Jaypee Institute of Information Technology University, 2009.

[9] H. P. Khandagale H.P.and Halkarnikar P.P." A Novel Approach for Web Page Change Detection System" June 2010, International Journal of Computer Theory and Engineering, 2(3), 793-8201.