International Journal of Computer Theory and Engineering, Vol. 2, No. 3, June, 2010
1793-8201

A Novel Approach for Web Page Change Detection System

H. P. Khandagale and P. P. Halkarnikar

Abstract— **Users are often not only interested in the current web page contents but also in changes in it. In this paper we describes web page change detection system based on Signature of Node corresponds to HTML pages. In this proposal first HTML document is filtered into XML structure document and then transforms XML pages to trees using DOM. The node signature comparison algorithm is developed to compare the trees of old web page and modified web page to find the changes in the web page. This system highlights changes of content i.e. deletion and addition of text and attribute changes i.e. font change, caption change, color change etc. and highlight the changed part in red color and displays to the user. This algorithm gets result faster as it does not search the sub tree if that sub tree does not have any changes.**

Index Terms— **Change monitoring, Signature of Node, Web Page change detection.**

I. Introduction

Web sites deliver information to the user in the form of web pages. The web is massively and constantly being updated with new pages that contain more and more information. Changes in web page will be classified as content changes. i.e. deletion and addition of fields in a table, attribute changes e.g. changes in the font, changes in the caption are termed as content changes. A user may visit certain Web pages repeatedly and is interested to know how each document has changed since the last visit. For example, a stock trader will be interested to know how the market is changing continuously every moment. The stock price can be changed. Similarly, to find certain items in the web pages which are changes frequently e.g. one could be interested in knowing the ranking of the hit songs. In this case changes in the ranking correspond to changes in the layout of the web page. Thus this system, helps to reduce the browsing time of the user and allows the user to find the items in web pages which change frequently. User may be interested in knowing the ranking of the hit songs. In this case changes in the ranking correspond to changes in the layout of the web page. Thus this system helps to reduce the browsing time of the user and allows the user to find the items in web pages which change frequently. In addition to above, the system can be helpful for versions and querying the past. The user may want to version a document or part of a web page and later need to ask a query about the past e.g., ask for the value of some element at previous time and to query changes e.g., ask for the list of items recently introduced in the catalog. The system is helpful for monitoring changes in a web page. This comparison process will be performed by generating the DOM trees for old web page and newly modified web page.

In this paper we describes web page change detection system based on Node Signature comparison mechanism corresponds to HTML pages. As part of the implementation, first HTML document is filtered into XML structure document and then transforms XML pages to trees using DOM structure. Then Node Signature algorithm is developed to compare the trees to find the change of text. The conversion of HTML pages to XML is done because XML is becoming the new standard for semi-structured data exchange over the Internet. The characteristics of XML, tree-structured (i.e. a collection of nodes) and self-descriptive, facilitate some form of automatic semantic integration. We assume only change in content and attribute of HTML tag so its corresponding change in the text and attribute node of xml document.

II. Literature Survey

Number of research papers was found that handled the design of efficient algorithms for detecting changes in Web pages. In [1], [2] and [3], various different algorithms are described for detecting changes in XML documents. The algorithm in [3] is based on finding and then extracting the matching nodes from the two trees that are being compared. From the non matching nodes, the change operations are next detected. Matching of nodes is based on comparing signatures (functions of node content and children) and order of occurrence in common order subsequences of nodes. The works in [1] and [2] transform the pages to trees according to the XML structure and use edit scripting to compare them. The strength of these algorithms lies in their low time-complexity, which is in the order of O (nlogn).

Recent work in change detection has focused on computing differences between flat files. The GNU diff utility and AT &Ts HtmlDiff [4, 6, and 7] are examples of this category. These examples use the LCS (Longest Common Subsequence) algorithm [8] to compare two plain text files or two HTML pages. This LCS works well for flat files; however, it fails to take into account the hierarchical structure information possessed by an XML document [5].

Manuscript received September 9, 2009

H. P. Khandagale, Department of Computer Science ,Shivaji University, Kolhapur , Kolhapur, Maharastra. India(email:k_hriday@yahoo.com).

P. P. Halkarnikar.,Department of Computer Science D. Y. Patil College of Engineering, Kolhapur, Maharastra. India(pp_halkarnikar@rediffmail.com).

In [9], Erwin Leonardi explained that change detection using ordered and unordered tree model approach is not suitable because lot of memory required keeping two versions of XML documents in the memory. In this paper author uses traditional relational database engine for detecting content changes of ordered large XML data. First it stores XML documents in RDBMS and then detects the changes by using set of SQL queries. This approach gives the better scalability and better performance. Different tree distance algorithms lead the foundation for change detection in structure document and XML. Out of that in [10] Tai presented tree-to-tree correction algorithm by defining the ordered mapping between two trees. This algorithm gives the best possible matching between two trees but with a runtime of O(n2h2), where n=max(n1,n2), h=max(h1,h2), h1=height of T1 and h2=Height of T2. XMLTreeDiff [11] computes difference between XML Documents based on heuristic similarity majors. It computes the hash values for the nodes of both the documents using DOM hash [12] and then reduces the size of the two trees by removing identical sub trees.

III. General Architecture of The Sysytem

The general architecture is shown in fig. 3.1. This system contains Comparator Module which contains Node Signature comparison algorithm which identifies the difference between two HTML pages as change in content and attributes. The architecture uses the stand alone model.

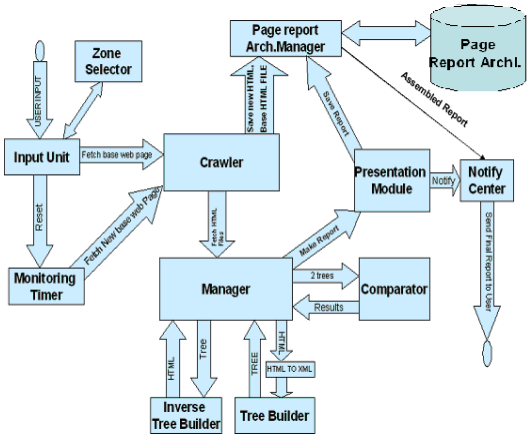

Fig. 3.1The General architecture of the web page change detection system.

There is a major path from the start state to the end state (denoted in the figure by ellipse). It corresponds to the steps taken to process a user request. When system starts, Option Form that interfaces to the Input Unit is available to user. From the Option Form user can select the Node Signature Comparison Form. After selection of the above form user will input the old Web Page and Modified Web Page. Afterwards, the Input Unit asks the Crawler to fetch the old Web page from achieve and Modified from site. These pages will be saved in Page Report Archive via Report Archive Page Manager and also passed to Manager. Now Manager Passes modified and the old HTML files to the HTML filter which converts HTML to XML files and passes to Tree Builder which returns two abstract trees corresponding to the two web pages. The Manager sends these trees to the Comparator which determines the differences between them. Afterwards Manager calls the Presentation Module and sends it the result. The Presentation Module makes a report out of results and saves it into the Page Report Archive .The Notification Center will assemble all the reports related to that web page, recompile a new report and sends it to the user. The Inverse Tree Builder accepts tree for modified page and converts tree to HTML page and gives it to browser for display as output.

A. Our Approach

We assume only text nodes are changed. Other types of nodes such as element and attribute nodes are not considered. This algorithm can identify a sub tree for further exploration while tossing out other sub trees if they do not show changes. Thus, exploration of the search space can be reduced. Our goal is at first to detect whether or not the two versions of web pages are identical. If not, we match each element value in the old version with its corresponding value in the modified version in order to detect value changes. This top down algorithm starts from the root nodes of the two versions by comparing signature values. Then the immediate children nodes of the root nodes will be compared. If a leaf node changes, the algorithm can detect changes not by inspecting the node itself but also its parent node, grand parent node and so on. For this, a signature of each node is constructed.

B. Signature Calculation

To reduce the number of node comparisons between two versions, we compare the node signature of a node in the old version with its corresponding node signature in the modified version. The algorithm first Assign Hash value for all child node of given tree that is text node using GetHash() function. Here the signature is the function of the hash value calculated from the contents of the node. Furthermore, the signature of interior node is simply summation of all its children node's signatures except the leaf node. Now the definition of signature is as follows

Suppose, X is a node in a DOM tree T,

Signature(X) = Signature(X1) + Signature(X2) +…. + Signature (Xn)

Where X1, X2 … Xn are the descendants of X.

If, X is an element associated with a text node (its value is V),

Signature(X) = Hash (V)

i.e.

Signature(X1) = Hash (V)

And

Signature $(X_n) = \sum_{x=1}^{x=n-1}$ sign(x)

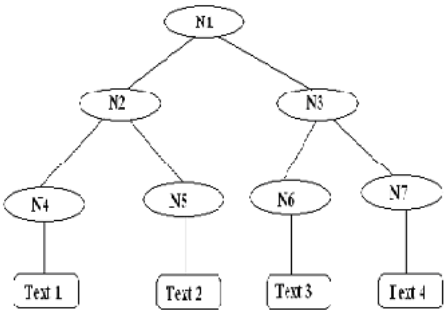

Fig 3.1 Typical tree for Signature Calulation

Pseudo code for Assign hash values is given as follows

Algorithm: Assign Hash value for all child node of given tree.
//This algorithm assigns Hash value for all child nodes of tree
Step 1: Input: Root node of tree
Step 2: For each child of current node
Step 3: obtain branch count of current node
Step 4: If (branch count==0)//i.e. current node is text node
……continue
Set ChildNode.tag = Hash value of text of child//Assign hash value to text node using GetHash() function
Step 5: Else
(Recursively call AssignHash() for present node) i.e. go to step2
Step 6: End.

Pseudo code of Signature Calculation given as

Algorithm: Calculate Signature
//This algorithm calculates the signature for all the tree nodes including root node
Input: Root Node
Step 1: For each tree node in the tree
Step 2: Check tag property of child node of current node
If child.Tag==null
goto step 1
Else
//Calculate signature of current node by adding the tag value of child to the tag value of current node.
currentnode.tag=currentnode.tag+child.tag
Step 3:End

C. Change detection

For the change detection, we first identify whether there is any change between Old Web Page and Modified Web Page. If the answer is yes, next we determine where this change occurs. With regard to first problem the signatures of two root nodes of Old Web Page and Modified Web Page are compared. If these signatures are same, there is no change between documents. If there is a change in any descendant node, the signature of this node will be changed as well as that of its immediate ancestor and thus that of the root node.

If signatures are different we need to address the problem of comparing nodes. For this, first children node's signature of the root node in the old web page will be compared with its corresponding nodes signature in the modified webpage. If these are same, sub trees rooted by these nodes from the web pages will be discarded. In other words, no further comparison will take place for these two nodes in the old web page and modified web page and their descendants. When the leaf nodes of the old web page and modified web page are reached and signatures are different, we will have arrived at the nodes where changes appear.

The pseudo code of Change Detection given as follows

Algorithm : Find changes
//This algorithm identifies the sub tree and text node of sub tree of new page where changes is occurred
Input: two trees T1 and T2.
Step 1://check for no change in page i.e. root nodes of T1 and T2 are equal
If(Signature of root node of T1 ==Signature of root node T2)……..continue
No change in new page
Goto End
Else
For each child node of T1 tree and for each child node of T2 tree
//Compare each child of T1 with its corresponding same child of T2
//compare child of T1 with child of T2 having some index.
If(child1.Index==child2.index)
If (Signature of child of T1 ! = Signature of child T2)
{ //check for text node
child1.Nodes.Count==0 && child2.Nodes.Count==0 of child1 and child2
Changes found. Text node i.e Text node T1 is change by Text node of T2
}Step2:End.

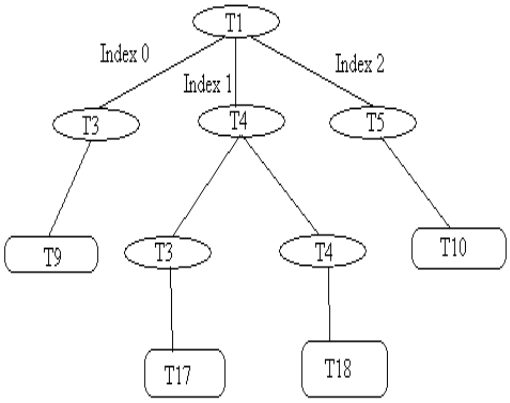

Fig.3.2 (a)

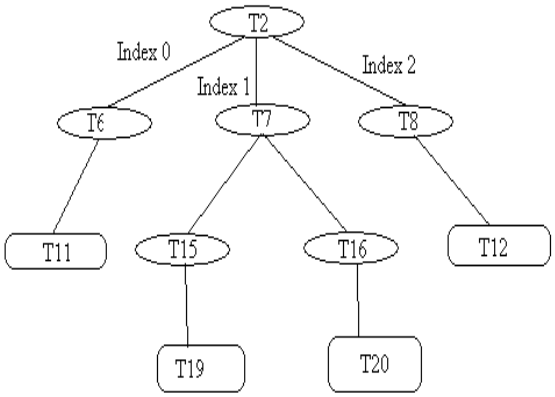

Fig. 3.2(b)

Fig 3.2 (a and b) Typical Trees for change detection

IV. Inverse Tree Builders

The Inverse Tree Builder rebuilds a highlighted Web page which shows the changed parts by using the information about the differences between the nodes of the

new tree and those of the old tree.

V. 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 5.1. The old web page and modified web page given as input to the node signature comparison interface produces results as shown in figure 5.2.

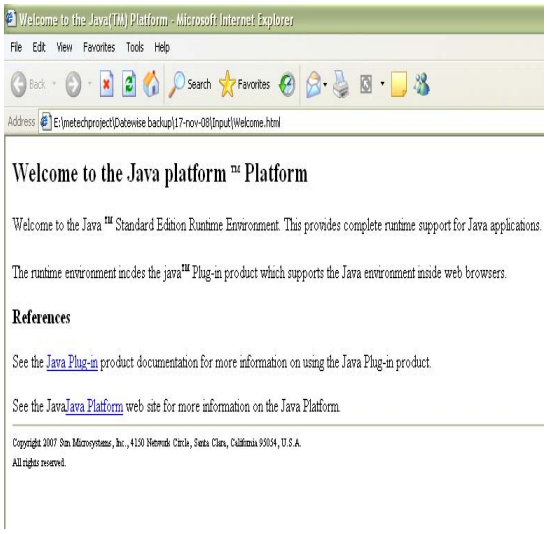

Fig. 5.1 Typical Web Page for Experiment

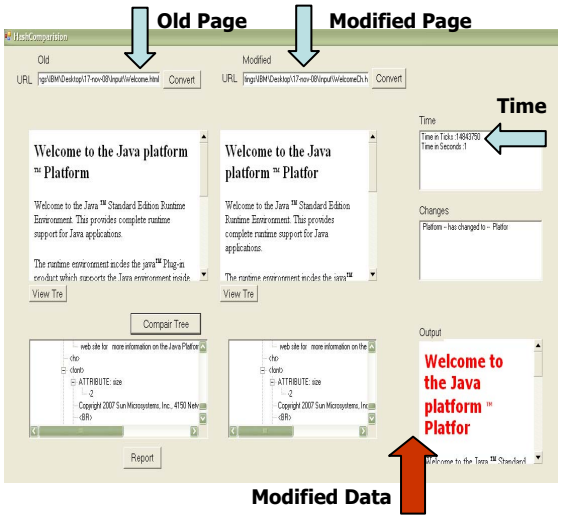

Fig 5.2 Change detection using Node Signature comparison

VI. Results

We have randomly changed less than 10 % of text of each web page and applied Generalize algorithm already developed by authors [22] and Node Signature algorithm. We measured the response time to detect the changes. Table 6.1 shows the list of readings. The response time of Node Signature is faster than Generalize algorithm.

TABLE 6.1 List Of Readings of Different Web Pages For Text Change

Nodes	Time	Algorithm
54	3.7	Generalize
90	4.6	Generalize
144	3	Generalize
211	3.6	Generalize
547	2.3	Generalize
574	1.9	Generalize
54	3	Node Signature
90	2.3	Node Signature
144	2.4	Node Signature
211	1.3	Node Signature
547	1.9	Node Signature
574	1.5	Node Signature

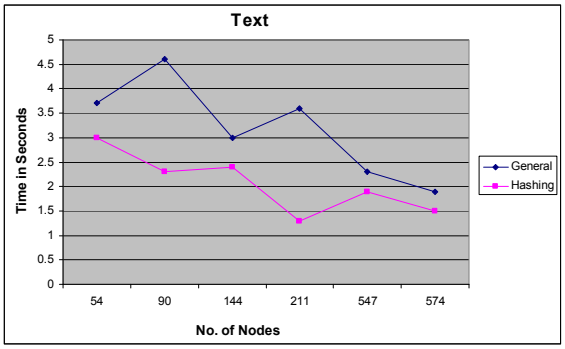

Fig 6.1Graph of performance results of different algorithms for change in Text

VII. Conclusion

The systems detect the changes in attribute, text and highlight the changed part in red color. This approach identifies the changes between the node signatures. It traverses fewer nodes in tree to detect changes between two web pages. Experimental results show that node signature is faster than Generalize tree algorithm.

The system is very useful for saving your countless browsing hours by keeping you up to date with what's new on web being monitor.

References

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

[2] Y. Wang, D. DeWitt, and J. Cai, "X-Diff: An Effective Change Detection Algorithm for XML Documents," Proc. 19th Int'l Conf. Data Eng., pp. 519-30, 2003.

[3] J. Jyoti, A. Sachde, and S. Chakravarthy, "CX-DIFF: A Change Detection Algorithm for XML Content and Change Visualization for WebVigiL," Data and Knowledge Eng., vol. 52, no. 2, pp. 209-230, 2005.

[4] E.Berk, "HtmlDiff: A Differencing Tool for HTML Documents", Student Project, Princeton University, http://www.htmldiff.com

[5] S.Chawate, A.Rajaraman, H.Garcia-Molina and J.Widom, "Change Detection in Hierarchical Structured Information", Proceedings of the ACM SIGMOD International Conference on Management of Data, Monteal, June 1996.

[6] F. Douglis, T. Ball, "Tracking and Viewing Changes on the Web", 1996 USENIX Annual Technical Conference, 1996.

[7] F. Douglis, T. Ball, Y.F.Chen, E.Koutsofios, "The AT&T Internet Difference Engine: Tracking and Viewing Changes on the Web", World Wide Web, 1(1):27-44, January 1998

[8] D. McArthur, "LSR Gene Expression RFI Response", Rosetta Impharmatics, http://cgi.omg.org/cgi-bin/doc?lifesci/99-08-11,August 1999.

[9] Erwin Leonardi, Sourav S. Bhownick, "Detecting Content Changes on Ordered XML Documents Using Relational Databases".
[10] K.C.Tai, "The tree-to-tree correction problem". Journal of the Association for computing Machinery, 26:485-495, 1979.
[11] F.P.Cubera , D.A. Epstein. "Fast Difference and Update of XML documents", Xtech, san Jose, March 1999.
[12] H. Mayurama, K. Tamora, "Digest value for DOM (DOM hash) proposal", IBM Tokyo Research Laboratory, http://www.tri.ibm.co.jp/projects/xml/domhash.htm,1998.
[13] Z. Bar-Yossef, T. Jayram, R. Krauthgamer, and R. Kumar, "Approximating Edit Distance Efficiently," Proc. 45th Ann. IEEE Symp. Foundations of Computer Sciences, pp. 550-559, 2004.
[14] Raymond Kosala and Hendrik Blockeel "Web Mining Research: A Survey"
[15] Latifur Khan, Lei Wang and Yan Rao "Change Detection of XML Documents Using Signatures".
[16] S.Chawathe and H. Garcia-Molina, "Meaningful Change Detection in Structured Data," SIGMOD Record, vol. 26, no. 2, pp. 26-37, 1997.
[17] S.-J. Lim and Y.-K. Ng, "An Automated Change-Detection Algorithm for HTML Documents Based on Semantic Hierarchies,"Proc. 17th Int'l Conf. Data Eng., pp. 303-312, 2001.
[18] Raihan Al-Ekram, Archana Adma and Olga Baysal "diffX: An Algorithm to Detect Changes in Multi-Version XML Documents."
[19] L. Liu, C. Pu, and W. Tang, "WebCQ—Detecting and Delivering Information Changes on the Web," Proc. Ninth Int'l Conf Information and Knowledge Management, pp. 512-519, 2000.
[20] H. P. Khandagale, P. P. Halkarnikar. "Web Page Change Detection System", National Conference on Algorithms, Fr. Conceicao Rodrigues College of Engineering, Mumbai, 16th and 17th May, 2008.
[21] Simon Robinson, Christian Nagel, Jay Glynn, Morgan Skinner, Karli Watson, Bill Evjen "Wrox Press-Professional C#3rdEdition
[22] P. J.Dietal, J. Liftfield "How to Write C# Program" Pearson Education , Low Prise Edition.

Pratap P. Halkarnikar received B.E. from Government College of Engineering, Pune in 1986. M.E. from Walchand College of Engineering, Sangli in 1993. Presently he is working as Assistant Professor in Department of Computer Science And Engineering at D.Y. Patil College of Engineering, Kolhapur. He is visiting Professor at P.G. center, Department of Technology, Shivaji University, Kolhapur. He is consultant to many industries for development of microcontroller based products. His interest lies in microcontroller based instrumentation, computer vision, data mining and web technology. He is member of ISTE.

Hridaynath P. Khandagale received B.E. in Computer Science and Engineering from D. Y. Patil College of Engineering, Kolhapur in 1997. M.Tech. (CST) from Department of Technology, Shivaji University, Kolhapur in 2009. Presently he is working as Assistant Professor in Department of Computer Science And Engineering at Bharati Vidyapeeth College of Engineering, Kolhapur. His interest lies in Application Programming, Data Mining and Web Technology.
He is member of ISTE.