COMPARATIVE ANALYSIS OF WEBPAGE CHANGE DETECTION ALGORITHMS

Srishti Goel*

Rinkle Rani Aggarwal*

ABSTRACT

Now a day's people are using internet actively for exchange of information across the world resulting in uploading of information on web pages and updating of new web pages very frequently. The contents of web page changes continuously & rapidly. Hence it becomes very difficult to observe the changes made in web pages and retrieve the original web pages.

To efficient retrieval and monitoring the changes made in web pages and compare the difference between refreshed page and old page efficiently that too in minimum browsing time, an effective monitoring system for the web page change detection based on user profile is needed. The web page change detection system can be implemented by using various Tools or Algorithms. In this paper, we will explain the application of various algorithms to detect the changes in web pages based on user intent.

*Department of Computer Science & Engineering, Thapar University, Patiala.

1. INTRODUCTION

With the increase in usage of internet for information the web pages are rapidly updated with more and more information. The user interest has changed from just seeking information to monitoring the changes. The users are interested to seek changes in particular section or portion and not on entire web page. Therefore there is a need to design a system helpful to the user to detect changes in entire page or particular section of web page. For example in a cricket match the statistics changes with every ball, in stock market the rates of shares changes frequently, news are updated rapidly. To understand these changes and to compare the changes with respect to specified time zone the web detection system is very important. This system helps the user to understand the changes effectively and efficiently in minimum browsing time.

2. CHANGE DETECTION

To detect the changes in a web page detection system we convert our HTML pages into the XML pages. Later these XML pages are converted into the tree structure. we can compare the tree structure of old web page and the modified web page. The two trees can be compared and the equal nodes and equal sub tree can be matched. The nodes and sub tree that don't matched are said to be different.

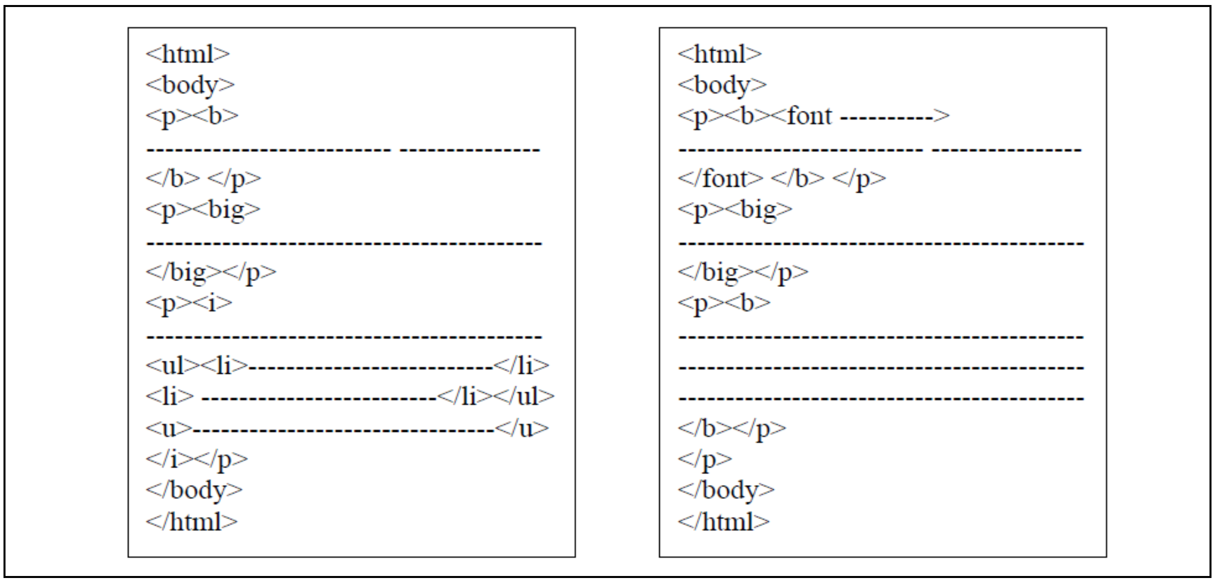
Classification of Changes in a Liable Web Page

The changes that can occur in web pages can be classified into following 4 major categories:

1. Structural Changes
2. Content level or semantic changes
3. Presentation or cosmetic changes
4. Behavioral changes

Structural Changes

[20]Sometimes the structure of web page is changed by addition/deletion of tags. Similarly, addition/deletion/modification in the link structure can also change the overall structure of the document (see Figure below). It may be noted that modification in existing link destinations may lead to altogether different documents. These types of changes are important to detect, as sometimes they often might not be visually perceptible.



```
<html>
<body>
<p><b>
------------------------- --------------
</b> </p>
<p><big>
-----------------------------------------
</big></p>
<p><i>
-----------------------------------------
<ul><li>-------------------------</li>
<li> ------------------------</li></ul>
<u>-------------------------------</u>
</i></p>
</body>
</html>
```

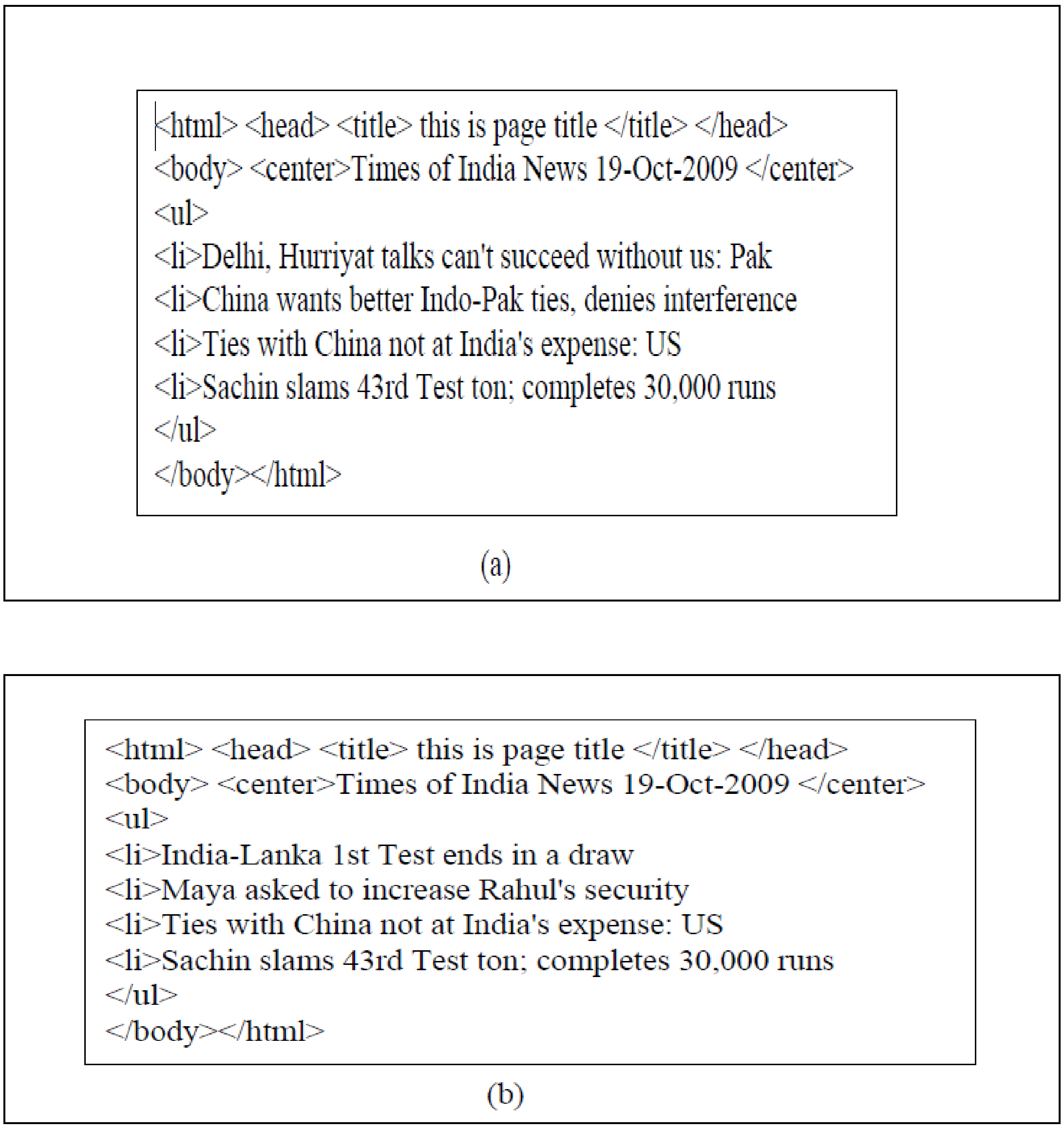
```
<html>
<body>
<p><b><font ---------->
------------------------ ----------------
</font> </b> </p>
<p><big>
-----------------------------------------
</big></p>
<p><b>
-----------------------------------------
-----------------------------------------
-----------------------------------------
</b></p>
</p>
</body>
</html>
```

Initial Version **Modified Version**

Figure 1: Structural changes in version of a web page[20]

Content Level or Semantic Changes

[20]These types of changes refer to the situation where the page's contents are changed from reader's point of view (see Figure). For example, a page created for a cricket tournament might be continuously updated as the tournament progresses. After the tournament ends, the page might altogether change its content and display information about award ceremony instead of scores. Similarly, all other web pages which are indirectly associated with the tournament such as the web pages containing the records of the players get modified accordingly to reflect the latest records.



```
<html> <head> <title> this is page title </title> </head>
<body> <center>Times of India News 19-Oct-2009 </center>
<ul>
<li>Delhi, Hurriyat talks can't succeed without us: Pak
<li>China wants better Indo-Pak ties, denies interference
<li>Ties with China not at India's expense: US
<li>Sachin slams 43rd Test ton; completes 30,000 runs
</ul>
</body></html>
```

(a)

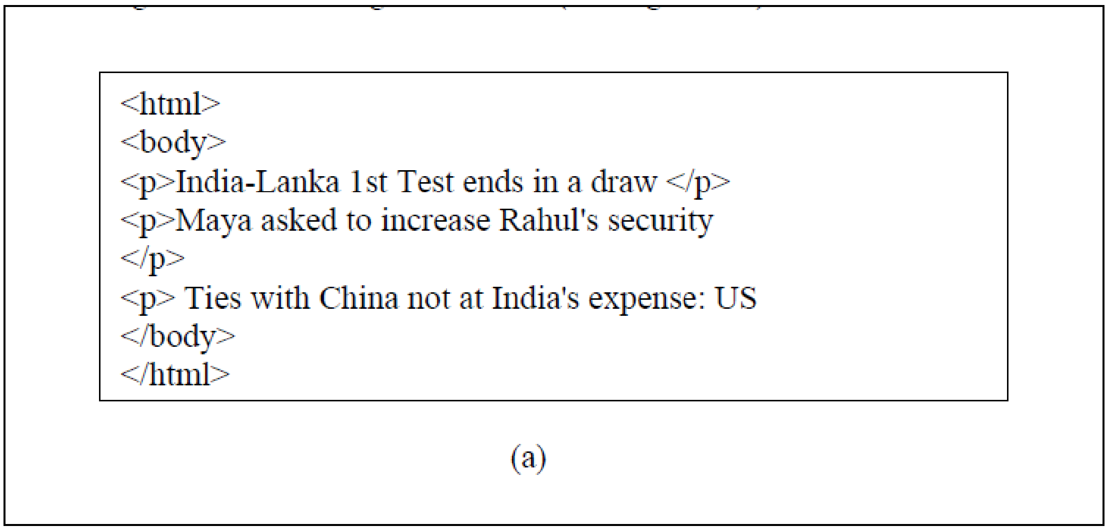
```
<html> <head> <title> this is page title </title> </head>
<body> <center>Times of India News 19-Oct-2009 </center>
<ul>
<li>India-Lanka 1st Test ends in a draw
<li>Maya asked to increase Rahul's security
<li>Ties with China not at India's expense: US
<li>Sachin slams 43rd Test ton; completes 30,000 runs
</ul>
</body></html>
```

(b)

Figure 2: Content/semantic changes (a) Initial Version (b) Changed version of a web page [20]

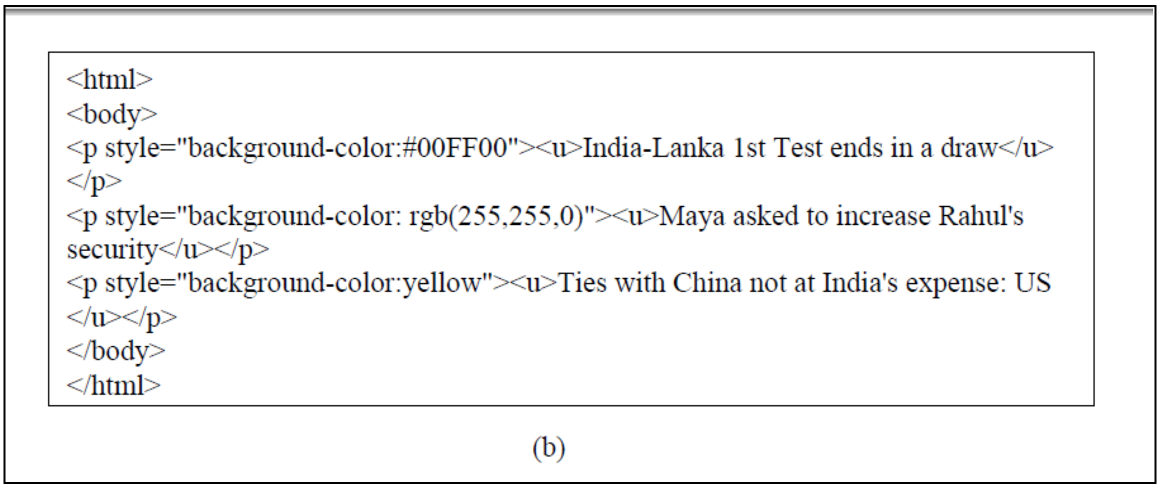
Presentation or Cosmetic Changes

[20]Under this category of changes only the document's appearance is modified whereas the contents within document remain intact. For example, by changing HTML tags, the appearance of document can change without altering its contents (see Figure).



```
<html>
<body>
<p>India-Lanka 1st Test ends in a draw </p>
<p>Maya asked to increase Rahul's security
</p>
<p> Ties with China not at India's expense: US
</body>
</html>
```

(a)



```
<html>
<body>
<p style="background-color:#00FF00"><u>India-Lanka 1st Test ends in a draw</u>
</p>
<p style="background-color: rgb(255,255,0)"><u>Maya asked to increase Rahul's
security</u></p>
<p style="background-color:yellow"><u>Ties with China not at India's expense: US
</u></p>
</body>
</html>
```

(b)

Figure 3: Presentation/ Cosmetic changes (a) Initial version (b) Changed version[20]

Behavioral Changes

[20]Behavioral changes refer to modifications to the active components present in a document. For example, web pages may contain scripts, applets etc as active components. When such hidden components change, the behavior 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 in figure 4:[6]. 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. 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.

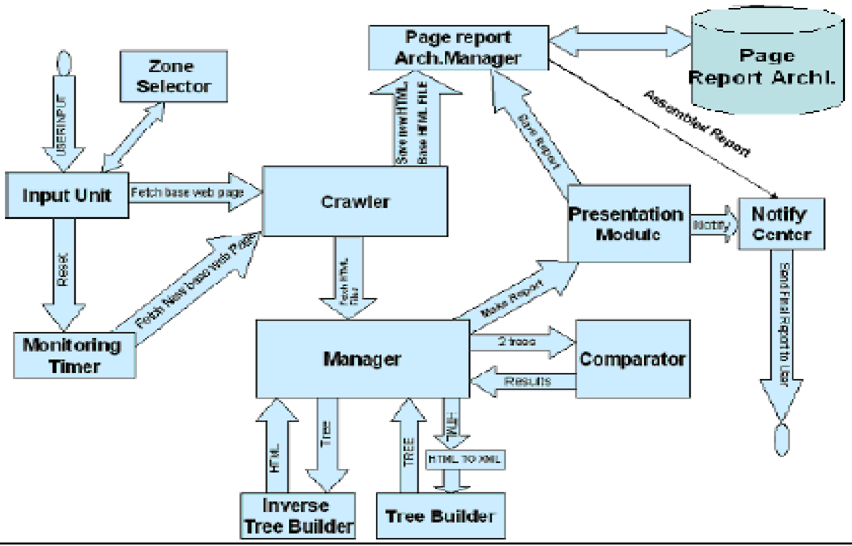


Figure 4:General Architecture Of Web Page Detection System[6]

4. CHANGE DETECTION ALGORITHMS

4.1 Novel approach

[6]We assume that only text nodes can be changed elements and attributes nodes can not be considered. This algorithm finds the subtree and deletes those trees that do not show changes. Firstly we find out whether the two trees are identical or not. If not, we try to find out the changes in two versions of the tree by comparing each and every element in old version and

the new version. This algorithm follow a top down approach it start with the root node by comparing the signature values of each node to its corresponding node in the two trees. To reduce the number of comparison we use node signature. Algorithm assigns the hash value to all child nodes in the trees the child nodes are basically the text nodes. Signature is mainly the function of the hash value calculated from the contents of the node. Signature of child node is basically the summation of its entire child node signature except the leaf node.

4.2 Level order traversal algorithm

This algorithm helps us to find the change in structure as well as in the content. [2]To find the changes in the structure i.e. to detect whether the new node has been added or deleted we use levels in the tree structure instead of traversing down the tree. And to find the content change RMS values of the ASCII values of the character are extracted from the web document. Changes in different versions of a web document can be effectively and efficiently extracted with the help of an algorithm. The change extractor performs the three things:

1. Document tree construction
2. Level order child enumeration
3. Finding the RMS values of the node.

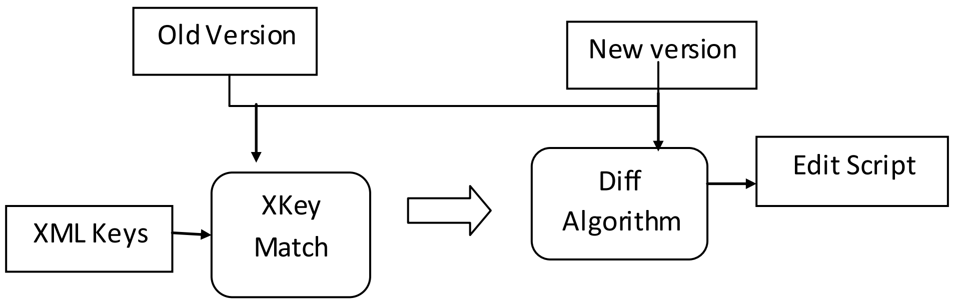
To evaluate the changes in the old version and the new version of a web document. We use the specialized set of arrays which shows the relationship between the parent and the child node.

4.3 La Diff

[14],[15]It is used for hierarchical structural information. It uses certain criteria two compare the two node of different version of a document in order to detect the changes between the two. Time complexity of this algorithm is $O(n*e+e^2)$ where e is the weighted edit distance between the two trees and n is the number of leaf nodes. Its complexity can be either linear or quadratic it can be linear in terms of size of the document and can be quadratic in terms of the number of changes between the two version.

4.4 X Key Match

[13] Algorithm for matching the two version of document using XML keys. This algorithm is used before diff algorithm so as to compare the two documents. The architecture of the system is shown in the figure.



Matched Nodes

Figure 5: X key match[13]

This algorithm is based on the construction of DFA from the set of XML keys. Matching is based on these keys only with a preorder traversal. Then we select the candidates for both the trees. These candidates can further be used for matching the two versions of trees. We use the function which is named matched to find the candidate having the same key values.

4.5 X Diff

[24]This algorithm can be used to detect the changes in two unordered tree. Edit script is used to convert one tree into another. Edit script is basically a sequence of edit operations. X diff algorithm uses three basic edit operations that are Insert, Delete and Update. X diff is used to find the minimum cost matching and also generates the minimum cost edit script. Various steps used in x diff are as follows:

1. Parsing and Hashing

   It parses the two XML document into the corresponding tree structure and assign a hash value to each and every node in the tree.
2. Matching

   This algorithm star with the root node and compare the hash values of the node in the two trees. If the hash value is same then the two trees are considered to be identical and if these values are different then minimum cost of matching is calculated for the two trees.
3. Generating minimum cost edit script

   Then it generates minimum cost of edit script based on the minimum cost of matching which is being generated in the previous step.

4.6 MH Diff

[24]It is a heuristic algorithm which is used to detect the changes in unordered structure documents. It uses edit script to transfer the tree into other. Exept the basic edit operations it uses two other operation move and copy, which result in higher quality of edit script. The Goal of this algorithm is to find the minimal edge cover, that corresponds to the minimum

cost edit script. This algorithm can be used only for small document and its worst case is O(n^3), where n is the number of nodes.

4.7 CX Diff ordered

[9]This algorithm is used to detect the customized changes to the content of XML document. To implement this algorithm first we extract the object in which the user is interested and then we apply the algorithm. In order to do so we detect the matching nodes between the two trees and from the non matching we will be able to detect the changed operations. Phases of this algorithm are as follows:

1. Object Extraction And Signature Computation
2. Filtering Unique Insert/Delete
3. Finding the common order subsequence

4.8 XML Tree Diff

XML Tree Diff is the set of java beans that can be used to efficiently differentiate and update DOM trees. [4] It work same as diff and patch work. To implement this algorithm we don't need any file format we can directly implement it on the DOM structure. It uses operations such as change node, delete node and insert node. It is well used for version management in two document and tree structured data.

4.9 MM Diff and XM Diff

[16], [17] Presents an external memory algorithm XMDiff (based on main memory version MMDiff) for ordered trees in the spirit of Selkow's variant. Intuitively, the algorithm constructs a matrix in the spirit of the "string edit problem", but some edges are removed to enforce that deleting (or inserting) a node will delete (or insert) the subtree rooted at this node. More precisely,

(i) diagonal edges exists if and only if corresponding nodes have the same depth in the tree

(ii) horizontal (resp. vertical) edges from (x,y) to (x+1,y) exists unless the depth of node with prefix label x+1 in D1 is lower than the depth of node y+1 in D2. For MMDiff, the CPU and memory costs are quadratic O(|D1| * |D2|).With XMDiff, memory usage is reduced but IO costs become quadratic.

4.10 IBM XML Diff and Merge Tool

[23]IT is created by IBM it is a java program which is used to detect the changes in the two versions of a XML document. This tool indicate each difference which are made in the document by symbols and colour coding.

4.11 3DM's matching Tools

[19]3DM's matching tool is used for performing 3 way merge and differencing of XML files.3DM tool is not only limited to insert, delete and update but also include move and copy operations.

4.12 XY Diff

[24]XY diff is a fast algorithm which support move operation as well as other XML features. It matches the large subtree found in both the XML documents. Matching of nodes is done according to the key attributes of the node. Then it first start with matching the large subtree and then the smaller subtree if the matching fails. If matching is succeeded then the parent node and the descendant of that particular node is compared. It supports insert, delete, update and move operations. It is not able to find the minimum edit script. It uses greedy approach because of which it does not provide optimal solution.

4.13 VM Tool

[22]It is a collection of java XML oriented tool and is available as a open source tool. It contains the diff and patch tools which can be used to match the two trees and automatically find the difference between them. Tool mainly focuses on the size of the XML file.

4.14 Diff XML

[21]It is a java based tool which is used to find the difference between the XML document. It operates on the hierarchical tree structure. The result is generated in a separate file this file can be a HTML file. The output report will contain the sections that were not matched. The tool match each and every node and their content and attributes and marks the difference with a color.

4.15 XMiddle

[1]It is a data sharing middleware .it is mainly used in mobile computing so that data can be share between the applications. Data is mainly encoded in XML. It allow other host on the network to access the data even when they are disconnected from the network. It also help to find the changes if there any. The use of XML as a underlying data structure enable XMiddle to find the difference and define reconcile policies for a particular document.

4.16 KF- Diff+

In this section we propose KF-Diff+, a highly efficient algorithm which need not to compute the hash signatures of all the nodes in the documents. [7]As mentioned before, the key property of KF-Diff+ is that the algorithm transforms the traditional tree-to tree correction into the comparing of the key trees. Since any two different nodes in the key tree should not have the same key path, the comparing between two trees will be more efficient.

There are two steps in KF-Diff+ as follows:

1) Parsing

KF-Diff+ parses DOC1 and DOC2 into trees T1 and T2.

2) Matching

The goal of this step is to find a minimum-cost matching between T1 and T2. Due to the space limitation, the generating of the edit scripts is omitted here.

4.17 XML Diff and Patch

[12]Microsoft created a tool [11] that is able to differentiate and patch two XML documents regardless if they are mapped into ordered or unordered tress. Moreover, it offers an option of ignoring whitespaces, XML comments and processing instructions, and offers some interesting options about the namespaces. It represents the differences with a XML Diff Language (XDL) that is called XML diffgram. However, it is not an open-source tool, it is not accompanied with any kind of documentation and it uses the Document Object Model [32]. Furthermore, it fails to compare files that differ dramatically and the online version of the tool is unable to handle files greater than 100KB.

4.18 Delta XML

Delta XML [3] is a commercial tool created by Monsell EDM Ltd, capable of comparing, merging and synchronizing XML documents. It is able to merge both ordered and unordered trees fast, where the size is up to 50MB. However, the state of the trees, either ordered or order less has to be explicitly stated. The XML documents have to be well-formed and with the same root element, whilst knowledge of the DTD or the Schema that it they follow is not required. One of its interesting features is that it supports both 2-way and 3-way merging.
Contrary to other tools, it only supports the insert, delete and update edit operations. The move edit operation is not implemented.

4.19 Tree Patch

Tree Patch is an open-source XML Diff and Patch Tool, designed. [10]It extended the Diff XML algorithm. Tree Patch describes changes using Extended Delta Update Language (EDUL). The EDUL output format describes the update, insert, delete and move operation.

4.20 Tree Match

Tree Match [10] is a fast tree pattern-matching algorithm for XML Query. The goal of Tree Match is to directly find all distinct matching of a query tree pattern in XML data sources. The algorithm is not applicable when detecting changes in XML documents because it is designed as a tree pattern matching algorithm and not as a tree-matching algorithm.

4.21 Bio Diff

Based on X-Diff, Bio Diff is an algorithm specially designed for detecting changes in genomic and proteomic data specified as XML[18]. It reduces the size of the set of biological data and focuses on the special semantics of the XML elements.

4.22 VI Diff

[25]It detects changes in a visual representation of a web page. It detects content as well as structural changes. Content changes include modification of the text, hyperlink or image. Whereas structural change include change in the visual appearance of the page. It can be implemented in various applications such as crawl optimization, archive maintenance, web changes browsing, etc.

Vi-DIFF consists of three steps:

(i) Segmentation,

(ii) Change detection

(iii) Generation of delta file

In addition to basic operations, it supports a move operation, if there is no structural change. However, it does not support yet the move on content changes when there are structural changes.

4.23 Optimized Hungarian Algorithm

[7]We use a system which transforms an HTML page into the XML document and then convert it into the tree like structure. Hungarian algorithm is used in the comparator module so as to compare two trees. We try to find the most similar sub tree in the new page and the old version of the page. Here we have used three optimizations over the Hungarian algorithm so as to compare the more efficiently.

Comparison of Various Change Detection Algorithm

	Refrences	Time complexity		Memory	Ordered/Unordered	Section
NOVEL APPROACH	6	Linear	?	?	Orderd	4.1
LEVEL ORDER TRAVERSAL ALGORITHM	2	Linear	?	?	Ordered	4.2
LA DIFF	14,15	Linear	$O(n*e+e^2)$	Linear	Ordered	4.3
X KEY MATCH	13	?	?	?	Both	4.4
X DIFF	24	Quadratic	$O(n^2)$	Quadratic	Unordered	4.5
MH DIFF	24	Quadratic	$O(n^2 \log n)$	?	Unordered	4.6
CX DIFF	9	?	?	?	?	4.7
XML TREE DIFF	4	Quadratic	$O(n^2)$	Quadratic	Ordered	4.8
MM DIFF	16,17	Quadratic	$O(n^2)$	Quadratic	Ordered	4.9
XM DIFF	16,17	Quadratic	$O(n^2)$	Linear	Ordered	4.9
IBM XML DIFF AND MERGE TOOL	23	?	?	?	?	4.10
3DM'S MATCHING TOOL	19	Linear	$O(n)$	?	Ordered	4.11
XY DIFF	24	Linear	$O(n \log n)$	Linear	Ordered	4.12
VM TOOL	22	?		?	Unordered	4.13
DIFF XML	21	Linear	$O(ne+e^2)$	Linear	Ordered	4.14
XMIDDLE	1	Linear				4.15
KF-DIFF+	7	Linear	$O(n)$	?	Both	4.16
XML DIFF AND PATCH TOOL	11,12	?	?	?	Both	4.17
DELTA XML	3	Linear	$O(\|x\| *D)$	Linear	Both	4.18
TREE PATCH	10	Linear	$O(ne+e^2)$	Linear	?	4.19
TREE MATCH	10	?	?	?	?	4.20
BIO DIFF	18	Quadratic	$O(n^2)$	Quadratic	Unordered	4.21
VI DIFF	25	?	?	?	?	4.22
OPTIMIZED HUNGARIAN ALGORITHM	7	Linear	$O(n^2)$	?	Ordered	4.23

5. CONCLUSION

From our study we concluded that to find a change in a old web page and a modified web page we design various algorithms apart from these algorithm various tools are also available in the market which can be used for this work. We have given the brief description of the algorithms and the various tools which can be used to detect the changes.

We have compared the algorithms and tools in terms of their complexity, memory and tree structure.

One of the algorithm introduces three running time optimizations that control the operations of the Hungarian algorithm.

Node signature algorithm was developed this approach identifies the changes between the node signatures. The systems detect the changes in attribute, text and highlight the changed part in red color.

REFERENCES

1. C. Mascolo, L Capra, S. Zachariadis and W. Emmerich" xmiddle: A Data-Sharing Middleware for Mobile Computing" journal wireless personal communication: An International Journal, Vol 21,Pp 77- 103,2002.
2. 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.
3. DeltaXML by Mosell EDM Ltd. Available at http://www.deltaxml.com. Accessed May 2003.
4. G. Cobena, S. Abiteboul, and A. Marian, "Detecting Changes in XML Documents," Proc. 18th Int'l Conf. Data Eng., pp. 41-52, 2002.
5. H. Artail *, K. Fawaz, A fast HTML web page change detection approach based on hashing and reducing the number of similarity computations, journal homepage: www.elsevier.com/ locate/datak, Data & Knowledge Engineering 66 (2008),pp 326–337
6. H. P. Khandagale and P. P. Halkarnikar A Novel Approach for Web Page Change Detection System in a International Journal of Computer Theory and Engineering, Vol. 2, No. 3, pp 364-367,June, 2010.
7. Kf diff + H. Xu, Q. Wu, H. Wang, G. Yang, and Y. Jia, "KF-Diff+: Highly Efficient Change Detection Algorithm for XML Documents", in Proc. CoopIS/DOA/ODBASE, pp.1273-1286, 2002.

8. I. Khoury, Student Member, IEEE, R. M. El-Mawas, Student Member, IEEE,O. El-Rawas, Elias F. Mounayar, and H. Artail, Member, IEEE ,An Efficient Web Page Change Detection System Based on an Optimized Hungarian Algorithm, in IEEE Transactions on knowledge and data engineering, VOL. 19, NO. 5,pp 599-612, MAY 2007.

9. 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.

10. J.T. Yao, M. Zhang "A Fast Tree Pattern Matching Algorithm for XML Query" IEEE/WIC/ACM International Conference on Web Intelligence, Pp 235-241. Sept. 2004

11. Xml diff and patch Komvoteas, K., 2003. XML Diff and Patch Tool. Available at: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.58.4650.

12. Xml tree and patch Microsoft Corporation. Available at http://apps.gotdotnet.com/xmltools/xmldiff/*accessed June 2003.*

13. P. Buneman, S. Davidson, W. Fan, C. Hara, and W. Tan"Keys for XML" Proceedings of international conference on Computer Networks, Vol39 No.5,Pp 473–487, Aug2002.

14. S. Chawathe, A. Rajaraman, H. Garcia-Molina and J. Widom, "Change Detection in Hierarchically Structured Information", Proceedings of the ACM SIGMOD International Conference on Management of Data, Montreal, Vol 25 NO.2, Pp 493-504, June 1996.

15. S. Chawathe, H. Garcia-Molina "Meaningful Change detection in structured data",proceeding in ACM SIGMOD International conference,Pp 26-37,May1997

16. MM diffCHAWATHE S., "Comparing Hierarchical Data in External Memory", *VLDB*, 1999.

17. MM diff[CHA 99b] CHAWATHE S. S., ABITEBOUL S., WIDOM J., "Managing Historical Semistructured Data", *Theory and Practice of Object Systems*, vol. 5, No. 3,Pp. 143–162,1999.

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

19. Tancred Lindholm, A 3-way merging algorithm for synchronizing ordered trees – the 3DM merging and differencing tool for XML, Master's thesis, Helsinki University of

Technology, Dept. of Computer Science, Sept. 2001. http://www.cs.hut.fi/~ctl/3dm/thesis.pdf, accessed May 2005.

20. Available at:www.shodhganga.inflibnet.ac.in/bitstream/1063...../14_chapter%205.pdf
21. Available at: https://sites.google.com/site/diffxml/
22. VM Tools by VM Systems, last release Feb. 2002 Available at: http://www.vmsystems.net/vmtools/, accessed May 2005.
23. XML Diff and Merge Tool by IBM alphaWorks, last update Mar. 2001,Available at: http://www.alphaworks.ibm.com/tech/xmldiffmerge, accessed May 2005.
24. 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.
25. Z. Pehliwan, M. saad, S.Gan, Carski "Vi-DIFF: Understanding Web Pages Changes".