Constructing and Implementing a New DOM-based Content Extraction Algorithm

Nang Kham Line Moong
Computer University, Mandalay
klmoong.aung@gmail.com

Abstract

The Internet explosion has made enormous Information sources published as HTML pages on the internet. However, there are many redundant pages as being known web pages noise on the Web. For instance, almost all dot com present a large amount of noise such as service channels, navigation panels, copyright and privacy announcement, advertisements, etc. Such noises can seriously harm Web Mining, Information retrieval and Information extraction. In this paper, a new algorithm is proposed and how it can be used to deal with Web page noises is also presented. The proposed algorithm matches DOM trees to classify which nodes are noises and which are contents and, after classification, cluster into their group respectively. Finally, only the content group is extracted from the page. The resulting contents are useful for both users and systems. The proposed technique leads to boost up the performance of Web Content Extraction.

1. Introduction

With the phenomenon growth of web, users have troubles in finding their needs and interests among a huge amount of Web data. Web mining appears to handle the problems. In Web mining, data preprocessing is the first step to eliminate the irrelevant data in web pages. Irrelevant data as being known web page noises are advertisement bars, navigational guides, decoration pictures, copyright and privacy notices, etc. Noises can be categorized into two types according to their level: Global noises and Local noises [2]. Global noises include mirror site, legal or illegal duplicated Web pages, old versioned Web pages. Local noises include [8]:

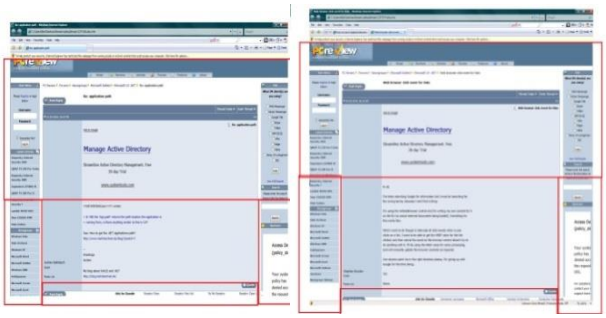
-Fixed Description Noise: site logos, decoration images and texts, copyright notices, privacy statement, etc.

-Service Noise: irrelevant services such as the weather, stock/market index, etc.

-Navigational Guidance: Directory guidance

-Advertisements.

This paper focuses on dealing with local noises. Most commercial web sites present their web pages with fixed layout or same presentation style. The most important fact is that noises are presented in same style and only their contents are different. Figure 1 shows two sample Web pages of the same web site. In this figure, only their content parts are different and others are the same.



page1 page2

Figure 1. Example different pages in same web site.

This paper presents our work that focuses on a new DOM-based content extraction algorithm. That is, the system is mainly concerned with the web content mining that extracts useful information from web pages. Although such web page decorations are good for web site owner and user's look and feel, it is non-valuable and also disturb in web mining process.

The main objective of the paper is to propose a new DOM-based content extraction algorithm and to present how it can be used to clean web pages noise. As the first step, HTML parser parsed the incoming web page and constructed DOM (Document Object Model) tree with preorder. Preorder traversal is arranged with root, left subtree and right subtree. It means, process all nodes of a tree by processing the root, then recursively processing all subtrees. In our paper, all nodes are classification and clustering by matching their DOM trees with each other. Finally, only content group is extracted from each page. Eliminate noisy information as the preprocessor of

information extraction application supports to get more precisely useful contents.

This paper is organized as follows. After this short introduction, Section 2 describes related work, Section 3 presents the basic ideas of the proposed algorithm, and Section 4 describes implementation of content extraction system. Finally, paper conclusion is described in Section 5.

2. Related work

Many web page cleaning approaches have been proposed to deal with web page noises. In [1], the approach works with a tree structure, Site Style to capture the common presentation styles and the actual contents of the pages. Information based measure support to determine which parts of SST represent noises or contents.

In [2], the proposed web page cleaning techniques is similar with [1]. First, create Compressed Structure Tree and use information-based measure to evaluate the importance of each node. The available importance nodes are assigning a weight to each word feature.

A DOM-based content extraction method has been proposed in [3] is implemented by filtering the DOM tree. When DOM tree is parsed through HTML parser, advertisement remover, link list remover and empty table remover are work for their responsibility.

Cleaning web pages for hypertext information extraction is described in [4]. Firstly, they construct DOM tree and mining noise data region from that tree by using predefined rules. Then noisy node discovery algorithm eliminates the mining data region. Because web sites have their own structure and presentation styles, cleaning noises by predefined rules are not convenient for all web sites.

S.-H. Lin and J.-M. Ho proposed dynamically selected the entropy-threshold that partitions blocks into either informative or redundant [5]. First, they partitions a page into several content blocks and calculates entropy value of the feature for defined the entropy value of the block.

D. D. C. Reis, P. B. Golgher and A. S. D Silva presented a domain-oriented approach to Web data extraction [6]. They based on the concept of tree-edit distance to evaluate the structural similarities between pages and extracted the relevant text massages discarding non-useful material such as banners, menus, and links.

3. Proposed algorithm

The proposed cleaning technique is based on matching web pages, which are represented by DOM trees. Firstly, HTML parser is used to parse the web page codes into corresponding tags. Then, the resulting tags are constructed into DOM tree. We give an overview of DOM tree in the following section. After that, DOM-based content extraction algorithm is widely discussed.

3.1 DOM tree

The Document Object Model (DOM) is a platform-independent and language-independent standard object model for representing HTML or XML and related formats [9]. With the Document Object Model, programmers can build documents, navigate their structure, and add, modify, or delete elements and content. Essentially, the DOM provides access to the structure of an HTML page by mapping the elements in that page to a tree of nodes. Each element becomes an element node, and each bit of text becomes a text node. Figure 2 shows a segment of HTML codes and its corresponding DOM tree. In this figure, body, table, b, and font are the element node and Bold and Font are the text nodes.

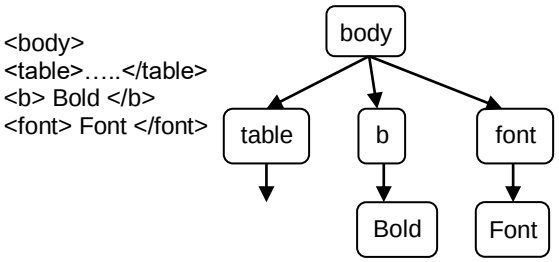


Figure 2. A sample DOM tree

There may be various DOM structures depending on the parser being used to build the DOM. In this paper, DOM tree is constructed with the help of HTML parser. The parser walks through all codes in the page and parsed into corresponding tags. The resulting tags are used in DOM tree construction.

3.2 Building DOM Tree

The system builds DOM tree with preorder traversal; processing with the root, left subtree and then right subtree. It means, it starts with the root node and walks through its entire child. Here is a series of steps, for example:

1. place the incoming node as current node
2. does this node have children? If so:
3. for each of the child nodes, go to step 1

This process is known as recursion, and is defined as the use of a function that calls itself. Each child is the same type of thing as the parent and can therefore be handled in the same way.

In the matching algorithm, it accepts two pages as parameter and outputs the content from the first page. Figure 3 shows the pages order in order to parameterize into the algorithm. In the figure, matching algorithm starts with accepted parameters: DOM1 and DOM2, and the resulted output are the

contents from DOM1. The system automatically saves the resulting content in text file format. Here are pairs of pages, for example, { (DOM1,DOM2), (DOM2,DOM3), (DOM3,DOM4), …… , (DOMn,DOM1) }. The process continues to parameterize pages in pair until all pages are processed.

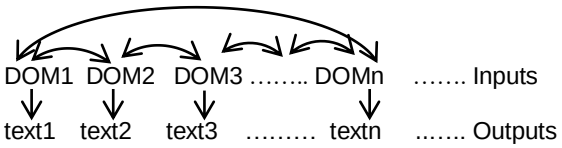
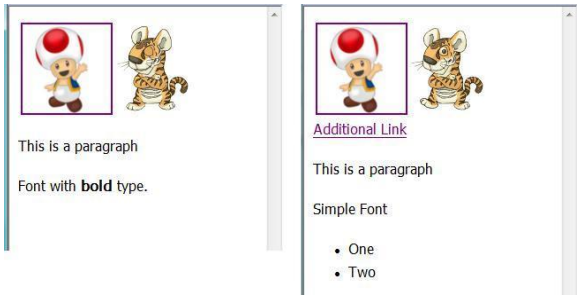


Figure 3. Pages order

3.3 DOM Tree Matching Algorithm

Before exhibiting the algorithm, sample pages are shown in figure 4 and its corresponding DOM trees are shown in figure 5 and 6. We aim to show their differences by comparing the two pages.



page1 page2

Figure 4. Example web pages

In figure 5 and 6, their differences are shown with shaded box, some tags are additional and some are different. In figure 7, the overall DOM-based content extraction algorithm is presented. The algorithm mainly matches the DOM trees in order to classify and cluster each node.

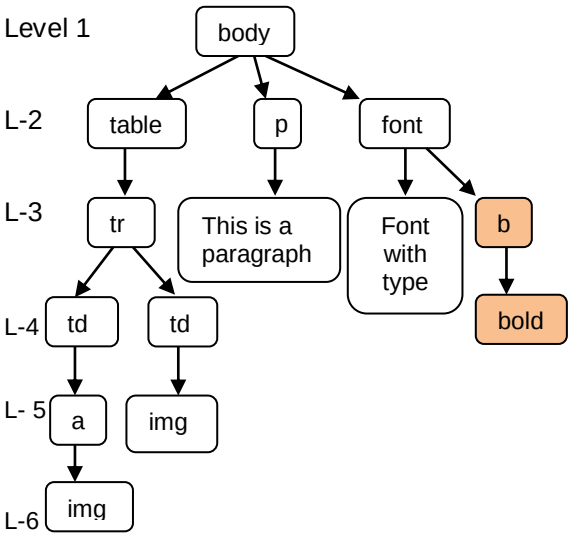


Figure 5. DOM tree for page1

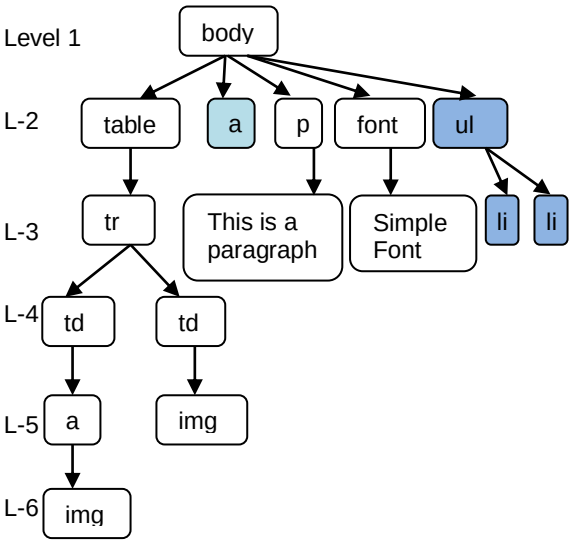


Figure 6. DOM tree for page2

3.4 Criteria for DOM Tree Matching

In matching process, all nodes classifying and clustering are nodes from DOM1. If the two pointed nodes have

Criteria 1: same level and same node then identify that node in DOM1 as noise.

Criteria 2: same level and different node then retrieved all nodes of DOM2 at that level and compared with the current node of DOM1. If they are equal, identify that node as noise. Otherwise, identify the current node of DOM1 as content.

Criteria 3: different level and different node then the system compares their level.

- **(a)** If DOM1's level is greater than DOM2, identify as content.
- **(b)** If DOM1's level is less than DOM2, then increase DOM2's pointer position.

In figure 7, the overall content extraction algorithm is presented. The algorithm is majoring worked with DOM tree node and node's level, which are essential in nodes classification.

Input: DOM1 **A** and DOM2 **B**
Output : Clustered nodes
Algorithm: DOM-based Content Extraction (A,B)

```
1. Begin
2.   Foreach TreeNode in A
3.     If A.Tag = B.Tag and A.Level = B.Level and
A.Text = B.Text then
4.     Add A's node into noise group
5.    Else
6.       If A.Level = B.Level
7.         Retrieve and store all nodes of B with current
A level
8.         Match A with available Array
9.         If node found
10           add A's node into noise group
11.        Else
```

```
12.         add A's node into content group
13.       End if
14.     Else if A.Level < B.Level
15.       Move B to next node
16.    Else
17.       add A's node into content group
18.    End if
19.   End if
20.  End for
21. End
```

Figure 7. DOM-based Content Extraction Algorithm

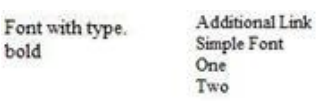
DOM trees are constructed and stored in array with the preorder traversal. Although the process moves with array index, all elements in array are tree nodes and also have DOM feature. Therefore, array indexes are used as pointers in comparison process.

The algorithm first targets to walkthrough all nodes in DOM1. It starts matching from body node that serves as the root node and continues until all nodes are classified. In two DOM trees at figure 5 and 6, both have the same nodes, same level and same text start from 'body' to all 'table' nodes, end at 'img' node. They all are classified as noise and clustered into noise group.

The algorithm increases the pointer position to continue the matching process. In this time, different nodes such as 'p' and 'a' are pointed, but both nodes are at the same level. In this case, algorithm retrieves all nodes of DOM2 at the same level with current node of DOM1. In figure 6, 'table', 'a', 'p', 'font' and 'ul' of DOM2 are retrieved. All retrieving nodes are compared to the current DOM1 node, 'p'. If match, DOM1's 'p' will be marked as noise. If not, it will be marked as content. We do nothing with DOM2 because it only used for matching, not for classified.

Once, DOM1 contains 'b' node but not in DOM2, the algorithm solves the problem by comparing their level. If DOM1 current node's level is greater than DOM2 current node's level, mark that node as content. If DOM1's level is less than DOM2's level, move DOM2's into next because we only focus on DOM1. From figures, 'b' is in DOM2 but not in DOM2, DOM2's current node is 'ul'. b's level is 3 and ul's level is 2. From the above definition, mark the node 'b' as noise. In this way, the algorithm walks through all DOM1 nodes and finally extracts clustered content.

In system implementation, text nodes are omitted. Instead, the text node's parent node is used as the leaf node in matching DOM trees.



page1's output page2's output

Figure 8. Result of two pages from figure 3.

Figure 8 shows the output of the first page and second page respectively. The page1's output must be "Font with bold type". Due to the order of the DOM tree node, the <font> tag is processed first and, after that, <b> tag is processed. Therefore, the result content structure is different from the input page.

4. Implementation

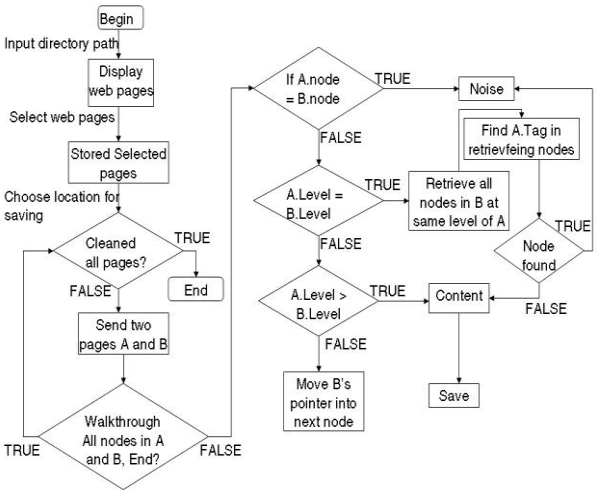


Figure 9. System Design

Figure 9 describes the implemented system design. In our system, user has to input directory path which contains a collection of web pages, choose the pages that he/she intend to process, and fill the location in order to save the resultant content files. The system saved the content files in text file format.

According to the input directory path, the system displays all pages in that supplied directory. When all requirements are completed, system starts the extracting process by using the proposed algorithm. Graphical user interface (GUI) is implemented in order to make our extractor easy to use. The system is implemented by using C# programming language. Figure 10 presents the main form of the system.

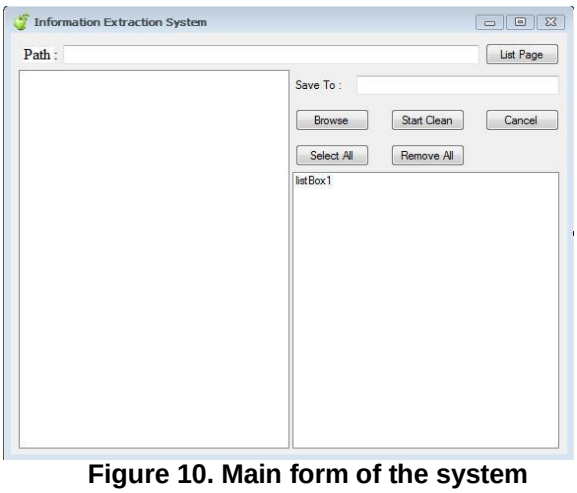


Figure 10. Main form of the system

In the GUI, “List page” button is used to represent all web pages in the input directory path. “Browse” is used for choosing the path in order to save the content file. “Start Clean” is used to start the extraction process. “Select All” and “Remove All” buttons are defined for supporting user when selecting web pages to process.

Figure 11 shows the form after the extraction process is worked. The extracted pages are listed and saved into the user predefined location.

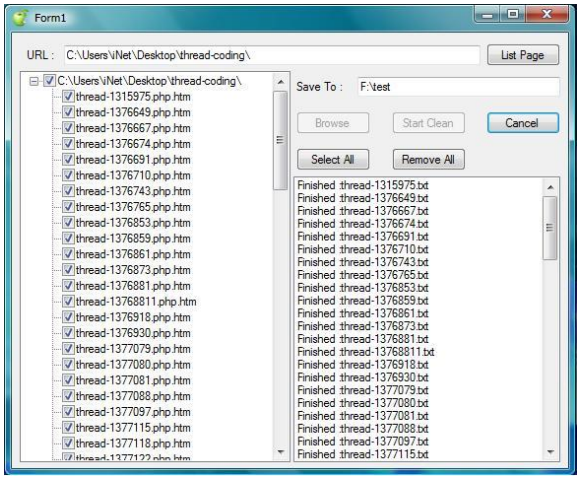


Figure 11. After extraction process

Depending on the type and complexity of the web page, the system can produce a wide variety of output. The algorithm performs well on pages such as news articles and mid-size to long informational passages such as education site, forums site and others, which contain a lot of message than figures.

Figure 12 and 13 show the example real world web pages. The system performs well on such page and the resulting content is satisfied.

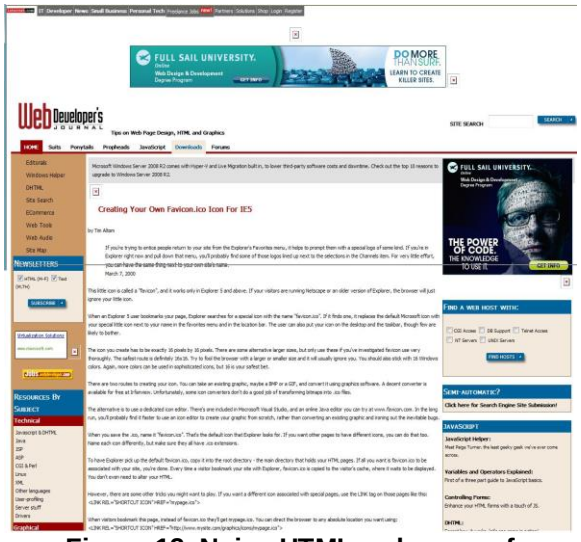


Figure 12. Noisy HTML web page of www.webdevelopersjournal.com

Figure 12 shows the real web page before being processed, that contains advertisement bars and side panels. Figure 13 presents after the extraction content of the page. The resulting contents are displayed in text editor. When saved in text file format, most of the resulting text is directly related to the content of the site, making it possible to use summarization and keyword extraction algorithms efficiently and accurately.

Site
promotionEngine
submissions

Compare PricesIP ServicesDesktop ComputersColocationTelevisions

Microsoft
Windows Server 2008 R2 comes with Hyper-V and Live Migration built in, to lower third-party software costs and downtime. Check out the top 10 reasons to upgrade to Windows Server 2008 R2.
Creating Your Own Favicon.ico Icon For IE5by Tim Altom

If you're trying to entice people return to your site from the Explorer's Favorites menu, it helps to prompt them with a special logo of some kind. If you're in Explorer right now and pull down that menu, you'll probably find some of those logos lined up next to the selections in the Channels item. For very little effort, you can have the same thing next to your own site's name.

March 7, 2000
This little icon is called a "favicon", and it works only in Explorer 5 and above. If your visitors are running Netscape or an older version of Explorer, the browser will just ignore your little icon.When an Explorer 5 user bookmarks your page, special icon with the name "favicon.ico". If it finds one, it replaces the default Microsoft icon with your special little icon next to your name in the favorites menu and in the location bar. The user can also put your icon on the desktop and the taskbar, though few are likely to bother. The icon you create has to be exactly 16 pixels by 16 pixels. There are some alternative larger sizes, but only use these if you've investigated favicon use very thoroughly. The safest route is definitely 16x16. Try to fool the browser with a larger or smaller size

Figure 13. Content Extraction from www.webdevelopersjournal.com page

4.1 Further example

Figure 14 and 15 show an example of a typical page from http://www.colorsontheweb.com/ and the content of that page, respectively. This page contains a lot of links structure, but our system performs well as show in figure 15.

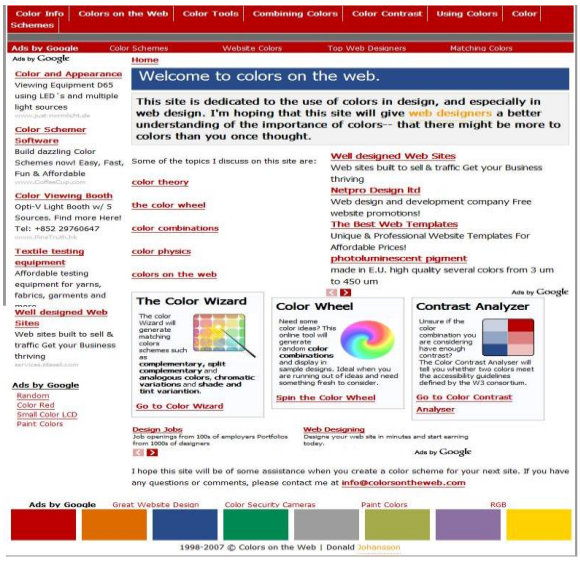
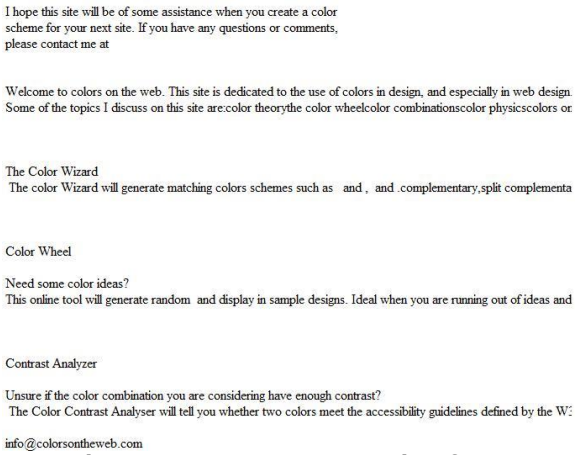


Figure 14. Noisy HTML web page of www.colorsontheweb.com


I hope this site will be of some assistance when you create a color scheme for your next site. If you have any questions or comments, please contact me at

Welcome to colors on the web. This site is dedicated to the use of colors in design, and especially in web design. Some of the topics I discuss on this site are:color theorythe color wheelcolor combinationscolor physicscolors or

The Color Wizard
The color Wizard will generate matching colors schemes such as and , and .complementary,split complementa

Color Wheel

Need some color ideas?
This online tool will generate random and display in sample designs. Ideal when you are running out of ideas and

Contrast Analyzer

Unsure if the color combination you are considering have enough contrast?
The Color Contrast Analyser will tell you whether two colors meet the accessibility guidelines defined by the W3

info@colorsontheweb.com

Figure 15. Content Extraction from www.colorsontheweb.com page

According to the resultant content of the implementation of the algorithm, our proposed cleaning algorithm makes effective and efficient results.

5. Conclusion

In this paper, DOM Tree Matching algorithm is proposed to clean web page noise. The system aims to support both users and systems whose works are only concerned with web page content. It also intended to improve web mining field such as web classification, web clustering and web personalization. Moreover, it is useful for students in collection their interest contents from web pages. According to the experimental implementation, the proposed cleaning algorithm is simple, fast and can automatically extract content. In addition, the limitation of the algorithm is that it does not work well for the pages with a huge number of figures and links. The limitations of the system are that it can only work with offline web pages, and cannot collect the content images. The effectiveness of our proposed algorithm and its implementation will be more obvious by combining with some tools or applications.

6. References

[1] L. Yi, B. Liu and X. Li, "Eliminating Noisy Information in Web Pages for Data Mining", in the proceedings of ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD-2003), Washington, DC, USA, August, 2003.

[2] L. Yi and B. Liu, "Web Page Cleaning for Web Mining through Feature Weighting", in the proceedings of Eighteenth International Joint Conference on Artificial Intelligence (IJCAI-03), Acapulco, Mexico, August, 2003.

[3] S. Gupta, G. Kaiser, D. Neistadt and P. Grimm, "DOM-based Content Extraction of HTML Documents", in the proceeding of the 12th World Wide Web conference (WWW 2003), Budapet, Hungary, May 2003.

[4] T. Win, K. N. Tun, "An Approach of Cleaning Web Pages for Hypertext Information Extraction" in International Conference on Computer Applications (ICCA 2008), Yangon, Myanmar, February 2008.

[5] S.-H. Lin and J.-M. Ho, "Discovering Information Content Blocks from Web Documents", in the proceedings of ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (SIGKDD'02), Edmonton, Alberta, Canada, 2002.

[6] D. D. C. Reis, P. B. Golgher and A. S. D Silva, "Automatic Web News Extraction Using Tree Edit Distance", in the proceedings of the ACM WWW conference (WWW 2004), New York, USA, 2004.

[7] B. Liu, R. Grossman, Y. Zhai, "Mining Data Records in Web Pages". UIC Technical Report, 2003.

[8] B. Liu, "Web Page Cleaning for Web Mining" in IJCAI-2003, KDD-2003. http://www.cs.uic.edu/~liub

[9] World Wide Web Consortium. Document Object Model specifications, January 2009. http://www.w3.org/DOM/