Web Page Segmentation and Informative Content Extraction for Effective Information Retrieval

Chaw Su Win
Faculty of Information and Communication Technology
University of Technology (Yatanarpon Cyber City)
Pyin Oo Lwin, Myanmar
chausuwin1982@gmail.com

Mie Mie Su Thwin
Myanmar Computer Emergency Response Team
Ministry of Science and Technology
Yangon, Myanmar
dmiesuthwin@mmcert.org.mm

Abstract— Internet web pages typically contain a large amount of non-informative content such as advertisements, search and filtering panel, headers, footers, navigation links, and copyright notices, etc. Such irrelevant information in web pages can seriously harm Web Mining. So the need of Informative Content Extraction from web pages becomes evident. Two steps, Web Page Segmentation and Informative Content Extraction, are needed to carry out for Web Informative Content Extraction. DOM-based Segmentation Approaches cannot often provide satisfactory results. Vision-based Segmentation Approaches also have some drawbacks. So this paper proposes Effective Visual Block Extractor (EVBE) Algorithm to overcome the problems of DOM-based Approaches and reduce the drawbacks of previous works in Web Page Segmentation. And it also proposes Effective Informative Content Extractor (EIFCE) Algorithm to reduce the drawbacks of previous works in Web Informative Content Extraction. In this paper Effective Informative Content Title Extractor (EICTE) Algorithm is also proposed to effectively extract the title of the informative content of the web page. The effective extraction results of the Proposed Algorithm, higher Precision and Recall can help for increasing the performance of Web Mining tasks.

Keywords: Informative Content Extraction, Main Content Extraction, Web Page Segmentation, Information Retrieval

I. INTRODUCTION

The rapid expansion of the Internet has made the web as a popular place for disseminating and collecting information. However, useful information on the web is often accompanied by a large amount of non-informative content such as banner ads, navigation bars, copyright notices, etc. Although such information items are functionally useful for human viewers and necessary for website owners, they can seriously harm automated information collection and mining on the web, e.g., Web Page Clustering, Web Page Classification, Information Retrieval and Information Extraction. So Informative Content Extraction from web pages becomes an interesting problem.

For Effective Informative Content Extraction from web pages the system needs to segment the web page into semantically related blocks correctly as the first phase. There are several kinds of methods for Web Page Segmentation. The most popular ones are DOM-based Segmentation, Location-based Segmentation and Vision-based Page Segmentation [6]. Actually DOM tends to reveal presentation structure other than content structure, and is often not accurate enough to discriminate different semantic blocks in a web page. In the Location-based Segmentation Approach, a kind of layout template cannot be fit into the layouts of all web pages. So, Vision-based Approaches have been used for Effective Web Page Segmentation. However, these Approaches also have some drawbacks and a good Vision-based Web Page Segmentation is urgently required. So this paper proposes Effective Visual Block Extractor (EVBE) Algorithm to overcome the problems of DOM-based Approaches and reduce the drawbacks of previous works in Web Page Segmentation.

The web page structure and layout varies depending on different content types and the presentation style designed by the web developer. Thereby informative content positions of the web pages differ in variety of websites. Even there might be some content in page view that are beside each other but actually in DOM tree they are not in the same level and same parent. So finding the informative content in this area needs complicated and costly algorithms [20]. Almost all algorithms have been proposed are tag dependent [4]. They could only look for informative content among specific tags such as <TABLE> or <DIV>. Algorithms that could simulate the behavior of a website user, Vision-based Approaches, have high probability for extracting informative content as result. So this paper proposes Effective Informative Content Extractor (EIFCE) Algorithm to automatically extract Informative Content Block from web page effectively. The Proposed Algorithm simulates the behavior of the website user and how the user finds the Informative Content Block in the page. The Proposed Algorithm is intended for improving the performance of Web Mining and Information Retrieval from web pages.

The rest of this paper is organized as follows. Section II introduces an overview on the literature review. In Section III, the background theory of the Proposed System is described. The Proposed System for Web Informative Content Extraction is described in Section IV. Section V proposes Effective Web Page Segmentation. Effective Web Informative Content Extraction is presented in Section VI. Section VII provides Experimental Results of the System. Evaluation of the System is presented in Section VIII. Finally, concluding remarks are expressed in Section IX.

© http://ijccer.org e-ISSN: 2321-4198 p-ISSN: 2321-418X Page 35
II. LITERATURE REVIEW

VIPS algorithm [6] uses visual cues to produce content structure from DOM structure and with this content structure it fills the gap between DOM structure and the conceptual structure of the web page. The algorithm uses obtained content structure and tries to simulate how the actual user finds the main content by blocking the page based on structure and visual delimiters. This algorithm mainly depends on visual separators although in some cases visual separators are misleading or ambiguous. It does many loops to reach its desire granularity.

Vision-based Page Segmentation (VIPS) algorithm needs some improvements in its most important part, Visual Block Extraction. So, Extended VIPS Algorithm [18] defines additional terms and detects visual cues for extending Visual Block Extraction part. In this technical report, deficiencies of VIPS algorithm are explained, and new rules are defined.

NEWIE [14] is proposed as a new approach to eliminate the noise in web pages for improving Informativ Content Extraction. The approach is DOM-based to eliminate the noise and defines the rules which are noise in web page based on the observation of the noise which are normally located in web page.

InfoDiscoverer Algorithm [4] is proposed to efficiently and automatically discover intra-page redundancy and extract informative contents of a page. The researchers concentrated on HTML documents with <TABLE> tags. Based on HTML tag <TABLE>, a page is partitioned into several content blocks. Based on DOM, a coarse tree structure is obtained by parsing an HTML page based on <TABLE>. Each internal node shows a content block containing one or more content strings as its leaf nodes. After parsing a page into content blocks, features of each block are extracted. Here features mean the meaningful keywords. After extracting features, entropy value of a feature is calculated according to the weight distribution of features appearing in a page cluster. Next step is calculation of entropy value of a content block. It is given by summation of its features entropies. The entropy of a content block is the average of all entropy values in that block. By using this, a content block is identified as informative or redundant. If the entropy value is higher than a threshold or close to 1, the content block is redundant as most of the block features appear in every page. If it is less than a threshold, the content block is informative as features of the page block are distinguishable from others.

Site Style Tree (SST) [7] considers non-content blocks as local noise in the web page. A tree structure is used to capture common presentation styles and actual contents of the pages in the given web site. A Style Tree can be built for the site by sampling the pages of the site. Each HTML page corresponds to a DOM tree where tags are internal nodes and the detailed texts, images or hyperlinks are the leaf nodes. A DOM tree can represent the presentation style of a single HTML page, but it is difficult to study the overall presentation style and content of a set of HTML pages. It is difficult to clean pages based on individual DOM trees. So a tree structure known as Style Tree is used for this purpose.

Two importance values, presentation importance and content importance, are used to find importance of an element node. The presentation importance is used to detect noises with regular presentation styles while content importance is used to detect those main contents of the pages that may be represented in similar presentation styles. Hence, the importance of an element node is given by combining its presentation importance and content importance.

CE [9], [10] considers all detailed pages of a website as pages with the same class. It runs a learning phase with two or more pages as its inputs and finds the blocks that their pattern repeats between input pages and marks them as non-informative blocks then stores them in storage. These non-informative blocks are mostly copyright information, header, footer, sidebars and navigation links. When using CE algorithm in actual world, it first eliminates non-informative patterns from the structure of its input pages based on the stored patterns in its storage for specific class of input pages. Then from the remaining blocks in the page it will return the text of block containing the most text length. CE needs a learning phase so it couldn’t extract the main content from random one input web page.

FE [9], [10] extracts the text content of block that has the most probability of having text so it will work fine in web pages that text content of main content dominates other types of content. In addition FE could return just one block of the main content, so K-CE [9], [10] is proposed in order to return k blocks with high probability of having the main content. Algorithm steps of K-CE and FE are the same except the last part. In K-CE, the algorithm final section sorts the blocks depending on their probability then it uses k-means clustering and takes high probability clusters.

CoreEx [11] is proposed as a heuristic technique for automatically extracting the main article from online news site web pages. It uses a DOM tree representation of each page, where every node in the tree represents an HTML node in the page. The amount of text and the number of links in every node are analyzed and a heuristic measure is used to determine the node (or a set of nodes) most likely to contain the main content. For every node in the DOM tree, two counts are maintained, textCnt holds the number of words contained in the node and linkCnt holds the number of links in or below the node. A node's score is a function of its textCnt and linkCnt. If two nodes reach identical scores, the node higher in the DOM tree is selected.

NIT [12] is a new method for information extraction from web pages. The method is based on statistical analysis of web page content and intended mainly for text corpus making. The NIT method is an automatic statistical-based algorithm using the web page structure for information extraction. It transforms the source document into the hierarchical structure DOM tree. Each DOM node represents one web page element. The NIT method is based on detection of most useful nodes in the DOM tree. The useful nodes are included into the extraction result and they represent an ideal plain text extract in the best case.

For extracting the title from an HTML document, Hu et al. proposed a machine learning approach with a large set of features based on both HTML tags and visual attributes [21].
A system for automatically extracting web article content and key components of the layout was proposed by J. Fan et al. [22]. The system not only identifies the content but also the structural roles of various content components such as title, paragraphs, images and captions.

III. BACKGROUND THEORY

A. Web Page Segmentation

Several methods have been explored to segment a web page into regions or blocks [2], [4]. In the DOM-based Segmentation Approach, an HTML document is represented as a DOM tree. Useful tags that may represent a block in a page include P (for paragraph), TABLE (for table), UL (for list), H1~H6 (for heading), etc. DOM in general provides a useful structure for a web page. But tags such as TABLE and P are used not only for content organization, but also for layout presentation. In many cases, DOM tends to reveal presentation structure other than content structure, and is often not accurate enough to discriminate different semantic blocks in a web page.

Another intuitive way of page segmentation is based on the layout of web page. In this way, a web page is generally separated into 5 regions: top, down, left, right and center [3]. The drawback of this method is that such a kind of layout template cannot be fit into all web pages. Furthermore, the segmentation is too rough to exhibit semantic coherence.

Compared with the above segmentation, Vision-based Page Segmentation (VIPS) excels in both an appropriate partition granularity and coherent semantic aggregation. By detecting useful visual cues based on DOM structure, a tree-like vision-based content structure of a web page is obtained. The granularity is controlled by the Degree of Coherence (DoC) which indicates how coherence each block is. VIPS can efficiently keep related content together while separating semantically different blocks from each other. Visual cues such as font, color and size, are used to detect blocks.

Each block in VIPS is represented as a node in a tree. The root is the whole page; inner nodes are the top level coarser blocks; children nodes are obtained by partitioning the parent node into finer blocks; and all leaf nodes consist of a flat segmentation of a web page with an appropriate coherent degree. The stopping of the VIPS algorithm is controlled by the Permitted DoC (PDoC), which plays a role as a threshold to indicate the finest granularity that we are satisfied. The segmentation only stops when the DoCs of all blocks are not smaller than the PDoC [8].

B. Informative Content Extraction

Informative Content Extraction is the process of determining the parts of a web page which contain the main textual content of this document. A human user nearly naturally performs some kind of Informative Content Extraction when reading a web page by ignoring the parts with additional non-informative contents, such as navigation, functional and design elements or commercial banners – at least as long as they are not of interest.

Though it is a relatively intuitive task for a human user, it turns out to be difficult to determine the main content of a document in an automatic way. Several approaches deal with the problem under very different circumstances. For example, Informative Content Extraction is used extensively in applications, rewriting web pages for presentation on small screen devices or access via screen readers for visually impaired users. Some applications in the fields of Information Retrieval and Information Extraction, Web Mining and Text Summarisation use Informative Content Extraction to pre-process the raw data in order to improve accuracy. It becomes obvious that under the mentioned circumstances the extraction has to be performed by a general approach rather than a tailored solution for one particular set of HTML documents with a well-known structure [19].

IV. PROPOSED SYSTEM FOR WEB INFORMATIVE CONTENT EXTRACTION

Figure 1 shows flow diagram for the Proposed System. It uses DOM tree of the input web page as its input and returns the Informative Content Block of the web page as its output. Five steps are needed to carry out for effective Web Informative Content Extraction: Cleaning the Web Page, DOM Tree Construction, Visual Block Extraction, Choosing Candidate Blocks and Finding Informative Content Block. In this paper these five steps are presented in two Phases, Effective Web Page Segmentation and Effective Informative Content Extraction. Another step, Informative Content Title Extraction is also proposed in Effective Informative Content Extraction Phase. The following sections demonstrate how to clean an HTML page, how to create the DOM tree of the input web page, how to extract semantic blocks of the web page, how to choose candidate blocks of the web page and how to extract the Informative Content Block and the Informative Content Title effectively. The terms used in the Proposed Algorithms of these steps are defined as follows.

Definitions of Terms in the Proposed Algorithms

DOM = DOM tree of the HTML page
BLT = Block Tree
BL = Block
RE = Root Element
CLS = Children
CL = Child
IFCBL = Informative Content Block
CBL = Candidate Blocks for the Informative Content Block
GP = General Parameters
DVaI = Distance Value of the Block
IFCTT = Informative Content Title
h1Title ~ h6Title and otherTitle = storages for respective string contents of H1 tags ~ H6 tags and the text content with valid visual prominence
titleCandidates = collection of candidate titles according to the priority h1Title ~ h6Title and otherTitle
candidateTitle = one of the candidate titles of titleCandidates
Title = the content of html <title> tag
ogTitle = the value of “content property” of <meta> tag with property = “og:title”
dcTitle = the value of “content property” of <meta> tag with property = “dc:title”
namETitle = the value of “content property” of <meta> tag with name = “title” or “DC.title”
infoTitle = variable which has boolean value to ensure whether it finds Informative Content Title or not

A. Cleaning the Web Page

Most web pages are not well-formed documents. They contain invalid tag structure such as there is an opening tag with no corresponding closing tag and vice versa. Some HTML tags are nested in wrong order and also some tags are mixed up. In order to construct the DOM tree of the input web page correctly, HTML file needs to be well-formed. Therefore these invalid tag structures are needed to clean before processing them. The CleanHTML Method for cleaning web page to construct the proper DOM tree effectively is as follows.

CleanHTML (HTMLpage)
for each HTML tag in page
begin
if the tag is missing or mismatched
then detect and correct this tag
else if the tags are nested in the wrong order
then correct the tag order
else if there are tags lacking close ‘>’
then fix this case
end if
end
end for

B. DOM Tree Construction

The Document Object Model (DOM) is a standard for how to access, change, add, or delete HTML elements. The DOM presents an HTML document as a tree-structure. By using DOM tree and its visual properties as a source of Web Page Segmentation instead of web page, more control can be achieved while segmenting the page.

Web pages are composed of HTML tags and their contents, such as text, images or hyperlinks etc. Each HTML page corresponds to a DOM tree where tags are internal nodes and the actual text, images or hyperlinks are the leaf nodes. The ConstructDOMTree Method that carries out to construct the DOM tree from the cleaned HTML document is as follows.

ConstructDOMTree (cleanedHTML)
begin
parse the cleanedHTML and construct the DOM tree
end

V. EFFECTIVE WEB PAGE SEGMENTATION

For Effective Web Page Segmentation, the following three steps:

A. Cleaning the Web Page
B. DOM Tree Construction and
C. Visual Block Extraction are needed to carry out.

In the Visual Block Extraction Phase, the Proposed System tries to overcome the problems of DOM-based Approaches and reduce the drawbacks of previous works by applying the proposed Effective Visual Block Extractor (EVBE) Algorithm and its efficient rules. EVBE Algorithm and its efficient rules are proposed in Web Page Segmentation Phase to help for getting effective result in Web Informative Content Extraction.
C. Visual Block Extraction

The Algorithm uses the DOM structure of the input web page and the visual properties of each DOM node as its inputs for effective extraction of semantic blocks from web page as shown in Figure 3. The Algorithm recursively traverses the input DOM tree in pre-order manner and returns the Block Tree and the General Parameters which contain general total value of specifications for text, width, height etc. of each block as its output.

```
Algorithm: Effective Visual Block Extractor (EVBE)
Input: DOM
Output: BLT, GP
Type: Recursive
begin
   ExtractBlock (RE, out BLT)
   if RE has CL then
     for each CL in RE’s CLS do
       if CheckValidity (CL) = True then
         RE := CL
         EVBE (RE, out BLT, out GP)
       end if
     end for
   end if
end
```

Figure 3: Effective Visual Block Extractor (EVBE) algorithm

The first line of the Algorithm applies ExtractBlock method by using the Root Element of the DOM tree as its input. It returns the Block Tree for the Root Element. If Root Element has Child, for each Child of Root Element the Algorithm checks whether the Child is valid for creating sub-Block Tree or not. If it is valid, EVBE Algorithm is used again for constructing sub-Block Tree of the Child node. Each node in Block Tree clusters one or more nodes from the DOM tree.

ExtractBlock Method

From the root node of the DOM tree, the process is started to extract semantic blocks from the DOM tree based on visual cues. It checks the input DOM node to determine whether it should form a new block or not. It makes this decision by checking the nature of the current node or the visual distance of the current node in contrast with its parent and in some cases with its sibling node. If the input DOM node is a separator node, the new block is created and the Algorithm flags its direct sibling node as the parent of new block. If the visual distance with its parent is valid, the current DOM node is added to the block of its DOM parent node. If the distance with its parent or in some cases with its sibling node is not valid, a new block is created and the Algorithm specifies the current DOM node as the parent of new block. The visual distance is the number of differences on the visual presentation styles of the DOM nodes such as width, height, font-size, background color, float, clear etc.

Sub-Block Tree Creation for Child Node

For each DOM child of the input element if the child is valid for creating sub-Block Tree, the recursive EVBE Algorithm is applied. The recursive algorithm makes sub-Block Tree for each child node and their subordinating child nodes.

EVBE Algorithm

Several algorithms have proposed DOM-based Approaches to segment a web page into semantic blocks. However, these algorithms cannot often provide satisfactory results. VIPS is a Vision-Based Algorithm to segment a web page into semantic blocks automatically. This algorithm mainly depends on visual separators although in some cases visual separators are misleading or ambiguous. It does many loops to reach its desire granularity.

The nature of EVBE Algorithm is different from VIPS Algorithm. The Proposed Algorithm mainly depends on the nature of the current node or the visual distance of the current node in contrast with its parent and in some cases with its sibling node for effective Web Page Segmentation. The definition of nodes and some important visual cues which are used for developing the efficient rules of the Proposed Algorithm are as follows.

Definition of Nodes

Inline Node: Nodes, which do not cause to a new line in a page, are inline nodes e.g., A, STRONG, BIG, EM, etc.

Line-break Node: Line-break nodes cause to a new line in a page e.g., DIV, P, TABLE, TR, UL, etc.

Invalid Node: Invalid nodes are those do not appear visually in the page, such as PARAM, SCRIPT, STYLE, TITLE, <!...>, !DOCTYPE, etc.

Partially Valid Node: These nodes affect the page layout only if they have a text node or any other visible children. Partially valid nodes consist of inline nodes and some of the line-break nodes such as FORM, HEAD, TABLE, DD, DT, etc.

Valid Node: Valid nodes are those appear in the page layout. They consist of partially valid nodes with visible children and
the remaining of the line-break nodes that do not include in partially valid nodes. These line-break nodes appear in the page layout, even if they do not have any valid child.

Text Node: The DOM node corresponding to free text, which does not have an html tag, is a text node.

Virtual Text Node:

1. Inline node with only text node children is a virtual text node.
2. Inline node with only text node and virtual text node children is a virtual text node.

Float: Some nodes can be grouped into the same block or divided into different blocks according to the nature of float values and in some cases together with the use of the width and clear properties of the nodes.

Clear: The clear property of the node is considered together with the float property when deciding which nodes should be grouped into the same block or divided into different blocks.

Width: The width property of the node is very useful to support other property of the node, such as float when making decision for creating blocks.

By using the above definitions and visual cues, the efficient rules for effective Web Page Segmentation are produced. Some basic rules of the Proposed Algorithm are as follows.

Rules of the Proposed Algorithm

Rule 1: Remove all the invalid nodes and partially valid nodes which do not have any child.

Rule 2: If a node has only text node or invalid children, no block will be extracted. Otherwise same rules are applied to each child.

Rule 3: If node has only one child,

1. If child is a text node or virtual text node, then no block will be extracted.
2. If child is line-break node, then same rules are applied to the child node.

Rule 4: If all the children are virtual text nodes of a node, node is put into block pool.

Rule 5: If a node contains a child whose tag is HR or any of valid nodes which has no child, then the node is divided into two as the nodes before the separator and after the separator. For each side of the separator, two new blocks are created and children nodes are put under these blocks.

Rule 6: If node is a table and some of its columns have different background colors than the others, divide the table into the number separate columns and construct a block for each piece.

Rule 7: If the first child of the node or the last child of the node is a DIV with no child and with nonzero height property and background-color property which is not white, then create two blocks: one of which for its parent node and the other for its parent previous or direct sibling node.

Rule 8: If a node has two adjacent children that the first child has float: left with or without clear; both property and the second child has float: left or right with clear: none property or without clear property and the total width of these two children is less than the width of the parent node, create a block for each adjacent child. Also create a block for the children without float value. Then same rules are applied to the children of each block.

Note: The parent node and two adjacent children may be DIV. Two adjacent children can have equal float value.

Rule 9: If a node has a child, whose at least one of border-top-width and border-bottom-width values is nonzero, divide this node into two blocks. Put the sibling nodes before the node with nonzero border into the first block and put the siblings after the node with nonzero border into the second block.

1. If child has only nonzero border-top-width, put the child into second block.
2. If child has only nonzero border-bottom-width, put the child into first block.
3. If child has both nonzero border-top-width and nonzero border-bottom-width, create a third block and put it between two blocks.

Rule 10: If a node has at least one of nonzero border-top-width and nonzero border-bottom-width values and both of the nonzero border-left-width and nonzero border-right-width values, create a block for the node.

Some rules of the Proposed Algorithm are similar with the basic rules from other Vision-based Algorithms. New efficient rules are presented in the Proposed Algorithm to get satisfactory results in Web Page Segmentation.

Effective Visual Block Extraction by Applying the Proposed Rules

Figure 4(a) shows the effective segmentation result on a sample page after applying the Proposed EVBE Algorithm. There are four semantic blocks in the page namely BL1, BL2, BL3 and BL4 marked by red rectangles. Here BL stands for block.

Figure 4(b) shows a part of the DOM tree to give an overview of the block extraction process. The visual block extraction process starts from the <body> node of the DOM tree. In the block extraction process, when the <div id = 'masthead'> node is met, it is checked whether it should be divided or not according to Rule 2. The visual distance between the current node and its children is valid and they are not divided into different blocks. And the algorithm checks the visual distance of the current node in contrast with its direct sibling node. The background color and some important visual properties of the two nodes are very different and the current node is extracted as a separate block BL1. In <div id = 'main'> node, there are two children, <div id = 'header'> and <div id = 'content'>. The children of <div id = 'header'> have valid visual distance with their parent. The node <div id = 'content'> has child <div id = 'main-content'>. In <div id = 'main-content'> node, there are two adjacent children <div class = 'block-a'> and <div class = 'block-b'>. The first child has float: left property and the second child has float: right
with clear: none property and the total width of these two children is less than the width of the parent node. So <div id = 'header'> node is extracted as a separate block BL2. And according to Rule 8, a block for each adjacent child is created as block BL3 and BL4. Therefore after applying EVBE Algorithm, four semantic blocks, BL1, BL2, BL3 and BL4 are extracted effectively as the semantic visual blocks of the sample page.

as shown in Figure 5. The first line of the Algorithm applies EVBE Algorithm for Effective Web Page Segmentation. It returns the Block Tree and General Parameters of each block as its output. Then the Algorithm applies ChooseCandidateBlocks function for choosing Candidate Blocks for the Informative Content Block. And then it applies the FindInformativeBlock method to detect the Informative Content Block from these Candidate Blocks. And it returns the Informative Content Block of the web page as result. Finally Effective Informative Content Title Extractor (EICTE) Algorithm is applied for extracting the title of the informative content effectively.

Algorithm: Effective Informative Content Extractor (EIFCE)

Input: DOM
Output: IFCBL, IFCTT
begin
EVBE (DOM, out BLT, out GP)
{ Final Computation for each total of GP }
ChooseCandidateBlocks (BLT, GP, out CBL)
{ Output: CBL }
FindInformativeBlock (CBL, GP, out IFCBL)
{ Output: IFCTT }
EICTE (IFCBL, Title, ogTitle, dcTitle, nameTitle, out IFCTT)
{ Output: IFCTT }
end

Algorithm: ChooseCandidateBlocks Function

Input: BLT, GP
Output: CBL
begin
for each BL in the BLT do
if GPs Valid (BL) then
CBL := BL;
end if
end for
end

A. Choosing Candidate Blocks

As shown in Figure 6, the inputs for ChooseCandidateBlocks function are the output of EVBE Algorithm, the Block Tree and General Parameters of each block. The output is the Candidate Blocks for the Informative Content Block. The method checks each block whether it is valid for choosing Candidate Blocks for the Informative Content Block. It is determined by checking the General Parameters of each important feature of each block. If the block met the least conditions to be chosen as candidate for the Informative Content Block, it is added to the Candidate Blocks list.

Algorithm: Effective Informative Content Extractor (EIFCE) algorithm

<table>
<thead>
<tr>
<th>(a) Effective segmentation result on a sample page</th>
</tr>
</thead>
<tbody>
<tr>
<td>(b): Corresponding DOM of the sample page</td>
</tr>
</tbody>
</table>

VI. EFFECTIVE INFORMATIVE CONTENT EXTRACTION

After segmenting the web page into semantic blocks correctly, the Informative Content Block of the web page can be extracted effectively by applying the next steps of Effective Informative Content Extractor (EIFCE) Algorithm

Figure 4 (a) and (b): Visual block extraction of the sample page

Figure 5: Effective Informative Content Extractor (EIFCE) algorithm

Figure 6: ChooseCandidateBlocks function

<table>
<thead>
<tr>
<th>(a) Effective segmentation result on a sample page</th>
</tr>
</thead>
<tbody>
<tr>
<td>(b): Corresponding DOM of the sample page</td>
</tr>
</tbody>
</table>

Figure 4 (a) and (b): Visual block extraction of the sample page

<table>
<thead>
<tr>
<th>effective segmentation result on a sample page</th>
</tr>
</thead>
<tbody>
<tr>
<td>corresponding DOM of the sample page</td>
</tr>
</tbody>
</table>

Figure 4 (a) and (b): Visual block extraction of the sample page

A. Choosing Candidate Blocks

Algorithm: ChooseCandidateBlocks Function

Input: BLT, GP
Output: CBL
begin
for each BL in the BLT do
if GPs Valid (BL) then
CBL := BL;
end if
end for
end
B. Finding Informative Content Block

The inputs for FindInformativeBlock function are the Candidate Blocks, the output of ChooseCandidateBlocks function and the General Parameters of each Candidate Block as shown in Figure 7. The output is the Informative Content Block of the web page. Firstly the method calculates the Distance Value DVal between the center of each Candidate Block and the center of the Page. The Distance Value is calculated by using Euclidean Distance Formula as in (1).

\[
\text{dist}((x,y),(a,b)) = \sqrt{(x-a)^2 + (y-b)^2} \quad (1)
\]

Then it compares the Distance Values of the Candidate Blocks and chooses two or three blocks with the nearest distances from the center of the Page. Finally from these chosen blocks the block with the largest width is returned as the Informative Content Block of the web page.

**Algorithm: FindInformativeBlock Function**

Input: CBL, GP

Output: IFCBL

begin
  for each BL in CBL do
    { Calculate DVal between the center of BL and the center of Page }
  end for
  for i=1 to j < no: of CBL do
    { Compare DVal of CBL }
    Choose two or three blocks with lower DVal }
  end for
  for j=1 to j < no: of chosen BLs do
    { Compare the width of the BLs }
    IFCBL := BL with the largest width }
end for

Figure 7: FindInformativeBlock function

C. Informative Content Title Extraction

After extracting the Informative Content Block of the input web page, Effective Informative Content Title Extractor (EICTE) Algorithm can be applied for extracting the title of the informative content effectively as shown in Figure 8. Title is a unique component of an article. Accordingly, it is often annotated with special html tags (H1 ~ H6) and given visual prominence. Especially, it is annotated with special header tags H1 and H2. Taking these properties into account, EICTE Algorithm includes two steps. In the first step of the Algorithm, ChooseCandidateTitles method, title candidates are selected according to the following criteria:

1. The top position of the text element must not below the center point of the Informative Content Block.
2. The font size must not be smaller than the font size of the main article text. Text elements with font size equal to that of the main article text are eligible only if they are either tagged with “H1” to “H6” or they are all bold.
3. The text content of the header tags H1 ~ H6 are preferable to be chosen as title candidates.

In the second step, FindInformativeTitle method, each candidate is taken into consideration according to its importance based on tag and visual prominence for choosing the correct title. The following HTML elements and visual features are used for detecting the Informative Content Title:

1. Title field: This is a text string delimited by the “<” and “>” tags in an html file.
2. Header tag “H1” to “H6”: The text elements under a DOM node with the header tags are given higher weight. “H1” and “H6” have the highest and the lowest weight, respectively.
3. The content property value of Meta tag: The value of the content property of <meta> tag especially with property = “og:title” or “dc:title” and the value of the content property of <meta> tag especially with name = “title” or “DC.title” are used for detecting Informative Content Title.
4. Font size: Title usually has large font size that makes it visually significant.
5. Position: The top position of the title used to be nearer with the top position of the Informative Content Block.

The inputs for EICTE Algorithm are the Informative Content Block of the input web page, Title, ogTitle, dcTitle and nameTitle. The output is the Informative Content Title of the web page. First the Algorithm applies ChooseCandidateTitles method for choosing candidate titles which can be the Informative Content Title of the input web page. Then it applies FindInformativeTitle method to detect the Informative Content Title from these candidate titles. The following sections demonstrate how to choose the candidate titles and how to detect the Informative Content Title effectively by applying the proposed ChooseCandidateTitles and FindInformativeTitle methods.

**Algorithm: Effective Informative Content Title Extractor (EICTE)**

Input: IFCBL, Title, ogTitle, dcTitle, nameTitle

Output: IFCTT

begin
  ChooseCandidateTitles (IFCBL, out titleCandidates)
  FindInformativeTitle (titleCandidates, Title, ogTitle, dcTitle, nameTitle, out IFCTT)
end

Figure 8: Effective Informative Content Title Extractor (EICTE) algorithm

C.A. Choosing Candidate Titles

The input for ChooseCandidateTitles is the Informative Content Block of the input web page as shown in Figure 9. The output is the candidate titles which can be the Informative Content Title of the input web page.
Algorithm: ChooseCandidateTitles Function

Input: IFCBL
Output: titleCandidates

begin
for each tag and text element in IFCBL do
    case tag of
    h1: Extract the content of <h1> tag.
        Add this string content to h1Title.
        Break.
    h2: Extract the content of <h2> tag.
        Add this string content to h2Title.
        Break.
    h3: Extract the content of <h3> tag.
        Add this string content to h3Title.
        Break.
    h4: Extract the content of <h4> tag.
        Add this string content to h4Title.
        Break.
    h5: Extract the content of <h5> tag.
        Add this string content to h5Title.
        Break.
    h6: Extract the content of <h6> tag.
        Add this string content to h6Title.
        Break.
    others: if the visual prominence of the text is valid for titleCandidates
            Add the text content to otherTitle.
    end if
end case
end for each
Merge all array lists as titleCandidates according to the priority h1Title ~ h6Title and otherTitle.
end

Figure 9: ChooseCandidateTitles function

For each tag and text element above the center point of the Informative Content Block, the method checks whether the tag matches with one of the header tags H1 ~ H6 or the text element has valid visual properties for choosing as candidate title or not. If the tag is <h1> tag, the method extracts the content of <h1> tag and adds this string content to h1Title which roles as an array list. And then the method reads the next tag for choosing title candidates. If this tag is not <h1> tag, the method checks whether the tag is <h2> or not. If the tag is <h2> tag, the method extracts the content of <h2> tag and adds this string content to h2Title which roles as an array list. And then the method reads the next tag for choosing title candidates. If the tag is not <h2> tag, the method continues its checking for candidate titles for all cases, H3 ~ H6 tags and the text element with valid visual prominence. If it finds a match, the string content is extracted and added to the respective h3Title or h4Title or h5Title or h6Title or otherTitle array list. If it doesn’t find a match, it omits the content of this tag or the text element from the consideration of candidate title. And then it checks the next tag for choosing candidate titles. The process of choosing candidate titles ends only after checking all tags which have top positions above the center point of the Informative Content Block. Finally the method adds all the candidate titles as titleCandidates according to the priority h1Title ~ h6Title and otherTitle. And it returns titleCandidates as the output of ChooseCandidateTitles method.

C.B. Finding Informative Content Title

Algorithm: FindInformativeTitle Function

Input: titleCandidates, Title, ogTitle, dcTitle, nameTitle
Output: IFCTT

begin
infoTitle := false
for each candidateTitle in titleCandidates do
    if candidateTitle = full or part of Title then
        IFCTT := candidateTitle
        infoTitle := true
        return
    else if candidateTitle= full or part of ogTitle then
        IFCTT := candidateTitle
        infoTitle := true
        return
    else if candidateTitle = full or part of dcTitle then
        IFCTT := candidateTitle
        infoTitle := true
        return
    else if candidateTitle=full or part of nameTitle then
        IFCTT := candidateTitle
        infoTitle := true
        return
end if
end for each
if infoTitle = false then
    Calculate the visual prominence of all titleCandidates.
    IFCTT := the one with highest visual prominence
    infoTitle := true
end if
end

Figure 10: FindInformativeTitle function

The inputs for FindInformativeTitle method are titleCandidates, the output of ChooseCandidateTitles and Title, ogTitle, dcTitle and nameTitle as shown in Figure 10. The output is the Informative Content Title of the input web page. For each candidateTitle in titleCandidates, the method checks whether the candidate title matches with full or part of Title or not. If it matches, this candidateTitle is decided as the Informative Content Title and assigns true value to infoTitle to ensure its find of Informative Content Title. And it returns this candidateTitle as the Informative Content Title of the input web page. If it doesn’t match with full or part of Title, the method checks whether it matches with full or part of ogTitle or not. If it matches, this candidateTitle is decided as the Informative Content Title and assigns true value to infoTitle. And it returns this candidateTitle as the Informative Content Title. If it doesn’t find a match, the method continues its detection for Informative Content Title for all cases, checking for a match with dcTitle and nameTitle. If it finds a match, this candidateTitle is decided as the Informative Content Title. And the method assigns true value to infoTitle, then returns
this candidateTitle as the Informative Content Title. If it doesn’t find a match with all these checkings, the value of infoTitle will still be false to ensure it doesn’t find the Informative Content Title. And the method continues its detection for Informative Content Title by calculating and comparing the visual prominence values of each candidateTitle in titleCandidates. Finally it returns one of the titleCandidates which has the highest visual prominence value as the title of the Informative Content.

VII. EXPERIMENTAL RESULTS

<table>
<thead>
<tr>
<th>No.</th>
<th>Address</th>
<th>Precision</th>
<th>Recall</th>
<th>F-measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td><a href="http://www.abcnews.com">www.abcnews.com</a></td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>2</td>
<td><a href="http://www.bbc.co.uk">www.bbc.co.uk</a></td>
<td>0.99</td>
<td>1.00</td>
<td>0.99</td>
</tr>
<tr>
<td>3</td>
<td><a href="http://www.cbsnews.com">www.cbsnews.com</a></td>
<td>0.98</td>
<td>1.00</td>
<td>0.98</td>
</tr>
<tr>
<td>4</td>
<td><a href="http://www.cnn.com">www.cnn.com</a></td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>5</td>
<td><a href="http://www.foxnews.com">www.foxnews.com</a></td>
<td>1.00</td>
<td>0.99</td>
<td>0.99</td>
</tr>
<tr>
<td>6</td>
<td><a href="http://www.fox2news.com">www.fox2news.com</a></td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>7</td>
<td><a href="http://www.themebright.com">www.themebright.com</a></td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>8</td>
<td><a href="http://www.washingtonpost.com">www.washingtonpost.com</a></td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>9</td>
<td><a href="http://www.latimes.com">www.latimes.com</a></td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>10</td>
<td><a href="http://www.huffingtonpost.com">www.huffingtonpost.com</a></td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>11</td>
<td><a href="http://www.thenion.com">www.thenion.com</a></td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>12</td>
<td><a href="http://www.indianexpress.com">www.indianexpress.com</a></td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>13</td>
<td>scitechdaily.com</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>14</td>
<td><a href="http://www.reuters.com">www.reuters.com</a></td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>15</td>
<td><a href="http://www.voanews.com">www.voanews.com</a></td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>16</td>
<td><a href="http://www.independent.co.uk">www.independent.co.uk</a></td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>17</td>
<td><a href="http://www.theatlanticpost.com">www.theatlanticpost.com</a></td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>18</td>
<td><a href="http://www.theregister.co.uk">www.theregister.co.uk</a></td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>19</td>
<td><a href="http://www.globalpost.com">www.globalpost.com</a></td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>20</td>
<td><a href="http://www.bloomberg.com">www.bloomberg.com</a></td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>21</td>
<td><a href="http://www.theguardian.com">www.theguardian.com</a></td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>22</td>
<td><a href="http://www.news.com.au">www.news.com.au</a></td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>23</td>
<td><a href="http://www.newyorker.com">www.newyorker.com</a></td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>24</td>
<td><a href="http://www.euronews.com">www.euronews.com</a></td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>25</td>
<td><a href="http://www.theaustralian.com.au">www.theaustralian.com.au</a></td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>26</td>
<td><a href="http://www.mizzimaburmese.com">www.mizzimaburmese.com</a></td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>27</td>
<td>first11.com</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>28</td>
<td><a href="http://www.mmtimes.com">www.mmtimes.com</a></td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>29</td>
<td>elevemynamar.com</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>30</td>
<td><a href="http://www.irrawaddy.org">www.irrawaddy.org</a></td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
</tr>
</tbody>
</table>

In this section EIFCE Algorithm is evaluated with the dataset from the following web sites which contain over 6,000 pages. As shown in Table 1 the Algorithm gains effective extraction results, higher Precision and Recall. And it also gains higher F-measure because of finer results of Precision and Recall.

VIII. EVALUATION OF THE SYSTEM

By applying the Proposed EVBE Algorithm, the blocks such as BL3 and BL4 can be extracted effectively. VIPS algorithm can segment them as separate blocks only if the Permitted Degree of Coherence (PDc) value is high. However, when the PDc value is high, it segments the page into many small blocks although some separate blocks should be a single block. It is unreasonable and inconvenient for any further processing. Actually the content nature of BL3 and BL4 is different and they should be segmented as separate blocks. The great rules of EVBE Algorithm can reduce the drawbacks of previous works and can help for getting finer results in Web Page Segmentation.

DOM-based Informative Content Extraction cannot often provide satisfactory results. CE needs a learning phase for Informative Content Extraction from web pages. So it couldn’t extract the informative content from random one input web page. FE and K-FE can identify Informative Content Block of the web page only if there is a dominant feature. The Proposed Algorithm can extract the informative content that is not necessarily the dominant content and without any learning phase and with one random page. It simulates the concept of how a user understands the layout structure of a web page based on its visual representation. Compared with DOM-based Informative Content Extraction, it utilizes useful visual cues to obtain a better extraction of informative content at the semantic level. The efficient rules of the Proposed EVBE Algorithm in Web Page Segmentation Phase can help for getting finer results in Web Informative Content Extraction. Experimental results show the Proposed System is highly effective for Web Informative Content Extraction.

IX. CONCLUSION

Web pages typically contain non-informative content, noises that could negatively affect the performance of Web Mining tasks. Automatically extracting the informative content of the page is an interesting problem. By applying the Proposed EVBE, EIFCE and EICTE Algorithms, the Informative Content Block and the Informative Content Title of the web page can be extracted effectively. The advantage of the Proposed Algorithm can be applied for improving the performance of Information Retrieval on the web. By applying the Proposed Algorithm as part of the Information Retrieval System, the relevance rank of the returned web pages can be improved. The retrieval accuracy can be increased, and the indexing size and retrieving complexity can also be reduced. Automatically extracting Informative Content Block from web pages can help for increasing the performance of Web Mining tasks.

REFERENCES


Proceedings of 2002 IEEE International Conference on Data Mining (ICDM'02), Maebashi City, Japan, December 2002.


