Combining DOM Tree and Geometric Layout Analysis for Online Medical Journal Article Segmentation

Jie Zou, Daniel Le and George R. Thoma
Lister Hill National Center for Biomedical Communications
National Library of Medicine
8600 Rockville Pike, Bethesda, MD, 20894
{jzou, daniel, gthoma}@mail.nih.gov

ABSTRACT
We describe an HTML web page segmentation algorithm, which is applied to segment online medical journal articles (regular HTML and PDF-Converted-HTML files). The web page content is modeled by a zone tree structure based primarily on the geometric layout of the web page. For a given journal article, a zone tree is generated by combining DOM tree analysis and recursive X-Y cut algorithm. Combining with other visual cues, such as background color, font size, font color and so on, the page is segmented into homogeneous regions. Evaluation is conducted with 104 articles from 11 journals. Out of 9726 ground-truth zones, 9376 zones are correctly segmented, for an accuracy of 96.40%. Segmenting the entire web page into zones can significantly expedite and increase the accuracy of the subsequent information retrieval steps.

Categories and Subject Descriptors
H.3.1 [**Information Storage and Retrieval**]: Content Analysis and Indexing – *Indexing methods*; H.3.6 [**Information Storage and Retrieval**]: Library Automation – *Large text archives*; H.3.7 [**Information Storage and Retrieval**]: Digital Library; I.7.5 [**Document and Text Processing**]: Document Capture – *Document analysis*.

General Terms
Algorithms, Performance, Design, Experimentation

Keywords: Document Object Model (DOM), Document Layout Analysis, HTML Document Segmentation, Web Information Retrieval

1. INTRODUCTION
MEDLINE® is the flagship database of the National Library of Medicine, containing over 14 million citations to the biomedical journal literature. It is important to have an efficient and reliable automatic system to extract bibliographic data for MEDLINE from online journal articles.

Many web information retrieval systems regard the web pages as the smallest undividable units. However, a web page usually contains various other contents besides the main topic, such as navigation panels, advertisements, banners and decorations. Even for the main topic of the web page, i.e., the article itself, it is usually necessary to divide it into different information zones, such as title, authors, affiliations, references and so on, for citation building purposes.

In this paper, we describe a systematic HTML web page segmentation algorithm, specifically designed to segment the online journals in order to expedite the subsequent information extraction processes in a reliable way.

We note that, similar to the traditional scanned documents, the most important cue to understand the semantic organization of an online journal article is the geometric layout of the web page. Therefore, unlike most of the existing methods, which mostly depend on HTML tags or DOM tree [2, 6, 10, 11], our approach heavily depends on the geometric layout of the web page. By combining DOM tree and traditional document geometric layout analysis, an HTML document can be represented by a zone tree model, which hierarchically organizes the regions of the web page into a tree structure. Depending on the requirements of subsequent information retrieval processes, other visual cues, such as background color and font attributes (size, color and face), can then also be combined to select a set of zone tree nodes, which appropriately segment the web page.

Many online journal articles are in PDF format. In order to standardize our input system and to minimize the number of modules for reading articles, we choose to convert them into HTML files by using an open source library, *PDFTOHTML*, [18] and then analyze the PDF-converted-HTML files. The PDF-converted-HTML files are quite different from regular HTML files. Figure 1 shows an example. It is the title of a journal article. (The whole page is shown in Figure 7.) Notice that it is broken into two <DIV> nodes. The positions of these two line texts are specified with absolute values (position:absolute). Besides this position information, there is not much we can use for segmenting the document. This is another reason for us to depend heavily on geometric layout of the web pages.

```
<DIV
style="position:absolute;top:126;left:91;height:25;width:697;"
><span class="ft0"><b>Identification of Hypertension-
Related Genes Through an</b></span></nobr></DIV>
<DIV
style="position:absolute;top:155;left:158;height:25;width:563;
"><span class="ft0"><b>Integrated Genomic-
Transcriptomic Approach</b></span></nobr></DIV>
```

Figure 1. A code snip of a PDF-Converted-HTML file.

JCDL'06, June 11–15, 2006, Chapel Hill, North Carolina, USA.

The rest of the paper is organized as follows. In Section 2, we review the existing HTML document segmentation algorithms and compare to ours. We also give a very brief review on the traditional scanned document layout analysis. We then introduce the zone tree model in Section 3. In Sections 4 and 5, the detailed algorithms for generating a zone tree and segmenting the page are described. Evaluation and error analysis are conducted in Section 6. Summary and conclusion constitute Section 7.

2. RELATED WORK

There are actually not many stand-alone reports on HTML document segmentation in the literature. Most of the previous studies treat the HTML document segmentation as a small component of a larger module in a web information retrieval system. Several authors argue that a complete and sound model for web pages may not be necessary for their specific tasks. This is not the case for online journal articles. Online journal article web pages usually can be logically divided into different zones. By modeling this zone structure, the article can be distinguished from its decoration and advertisement parts, and the article itself can be further segmented into regions of title, author, affiliation, abstract, sections, acknowledgement, references, and so on. Subsequently, the bibliographic data, including title, author, affiliations, grant number and databank, comment in and comment on, can be much more reliably and efficiently collected.

We believe that, just as several well-known models have already been proposed for analyzing scanned documents, e.g. X-Y tree [13], Docstrum [15], Block Adjacency Graph (BAG) [9], etc., a complete and sound semantic model for online journal article HTML pages is critical for efficient and reliable analysis.

A straightforward approach for segmenting web pages is to use tag information. Usually, a small set of tags serve as the segment indicators. In [6], four types of tags, including <P>, <TABLE>, <LI>/<UL> and <H1>~<H6>, are used to detect four major types of segments: paragraph, table, list and headings, respectively. In [11], only <TABLE> tag is used to partition a page into several blocks. Similarly, in [2, 10], several simple tags, such as <P>, <TABLE> and <UL> are chosen to divide the web page for subsequent conversion and summarization.

A Function-based Object Model (FOM) is proposed in [5]. FOM attempts to understand author's intention by identifying object functions. By grouping objects based on the identified functions, a hierarchical FOM structure is then generated for the page. However, the functions and properties of the objects and the grouping rules are hard to define. Most importantly, FOM is designed for website adaptation, and not for semantic understanding of the web page.

The VIPS (VIsion-based Page Segmentation) algorithm [3, 4] is the most similar work to ours. In VIPS, a tree structure is used to model the page. Each node corresponds to a block in a page, and has a value to indicate the Degree of Coherence (DoC). The DOM tree is analyzed from root to leaves and the DOM nodes are divided based on their spatial layout and several other visual cues, such as color. The process continues until the DoC of the leaf tree node meets the predefined DoC.

In VIPS, assigning the degree of coherence needs to balance many incomparable features and is a difficult task. We concentrate on online journal articles, where the geometric layout is the dominant cue for semantic organization. Furthermore, in PDF-Converted-HTML files, geometric layout is the only information we can utilize for segmentation. Therefore, we choose to avoid the difficult task of trading off several incomparable features, but depend only on geometric layout to determine the relationship between zones.

In VIPS, the DOM tree is expanded layer by layer, and a set of complicated heuristic rules is defined to decide whether to divide a particular DOM node. In our work, we choose to break the DOM nodes down to the deepest level, i.e., collecting only the naturally undividable DOM nodes. (The details are explained in the following sections.) These undividable DOM nodes are leaf zones. The zone tree is then created by analyzing the geometric layout of the leaf zones. This approach makes the zone tree more independent from the DOM tree, i.e., the zone tree structure depends more on the leaf nodes' geometric relationship than their positions in the DOM tree. This choice is based on our understanding that DOM tree follows exactly the original HTML code and therefore is a model at a syntactic level, mostly for displaying and manipulating HTML pages. On the other hand, the geometric layout is a very powerful cue for grouping related information. Therefore, although we pay close attention to the DOM tree, we try to minimize its effect on the zone tree. Tracing down directly to the deepest leaf zones also increases the speed of the algorithm.

In brief, VIPS is designed to handle any kind of web pages, but our work is only concerned with online journal article web pages. Online journal article pages allow us to concentrate mostly on geometric layout of the web page. Our algorithm, thus, is able to avoid many ad hoc rules, and adopt a clean model, which makes the algorithm more efficient and reliable.

Geometric layout analysis on scanned documents has been extensively studied and documented in the literature. Most of the algorithms can be categorized as following either a top-down or bottom-up approach. Top-down algorithms recursively divide a whole page into smaller zones. The process terminates when some criteria are met. Typical top-down methods include the X-Y cut [13, 7], shape-directed-covers-based algorithms [1] and several others. Bottom-up algorithms start with the image pixels, and cluster them into connected components, then into words, lines and finally zones. Typical bottom-up methods include Docstrum [15], Block Adjacency Graph (BAG) [9], and several others. Hybrid methods combining split and merge strategies are also proposed in [16, 8]. A review on scanned document image analysis is given in [14].

We choose the classic recursive X-Y cut method in our segmentation algorithm. X-Y cut, primarily based on the gaps between adjacent blocks, is a simple and efficient algorithm [13, 7]. The major drawback is that the algorithm is sensitive to skew and noise. However, it is not a problem for online pages. The bounding boxes of DOM nodes are straight and clean.

3. HTML DOCUMENT MODELS: DOM TREE AND ZONE TREE

In order to handle the structured document, such as HTML and XML, more efficiently and consistently, the World Wide Web Consortium (W3C) published the Document Object Model (DOM) specification. DOM is a set of platform and language

independent application programming interfaces (APIs) that describe how to access and manipulate information stored in a structured HTML or XML document. [12, 17]

HTML DOM is in a tree structure, usually called an HTML DOM tree. Figure 2 illustrates a simple HTML document and its corresponding DOM tree. We are interested only in the <BODY> node and its offspring. In this example, <BODY> node has three children: element nodes <B> and <I>, and text node #and. Element node <B> has a text node child #bold, and element node <I> has a text node #italic. Following the DOM convention, we use <> to indicate element node, and use # to indicate text node.

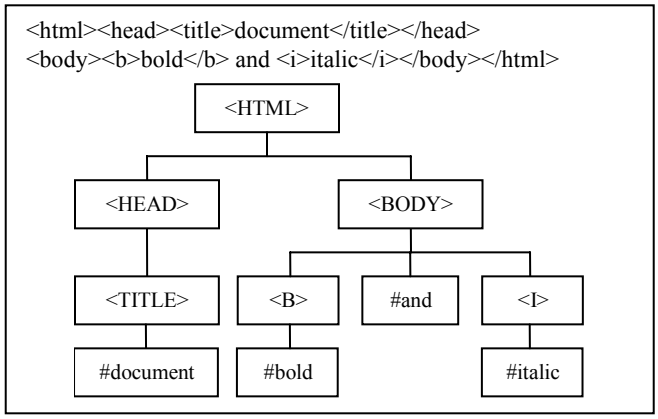


Figure 2. A simple HTML document and its DOM tree.

In order to make the following discussion easier, based on the HTML specification from W3C, we categorize the HTML DOM nodes as follows:

Insignificant node: A type of node that cannot be seen through the browser. This type includes nodes containing only space, line feed, carriage return, or   characters and nodes that do not occupy any space on the page, such as comment nodes, hidden nodes, (e.g., <INPUT type=hidden>) and so on, with exception of
 nodes, which are line-break nodes.

Inline node: This type of node does not introduce line breaks. A complete inline node tag list in our algorithm includes: <A>, <ACRONYM>, <ABBR>, <B>, <BIG>, <CITE>, <CODE>, <DEL>, <DFN>, <EM>, <FONT>, <I>, <IMG>, <INPUT>, <INS>, <NOBR>, <KBD>, <Q>, <SAMP>, <SMALL>, <SPAN>, <STRONG>, <SUP>, <SUB>, <TT>, <U>, <VAR>.

Line-break node: A node which is neither insignificant nor inline.

DOM tree is a well defined HTML document model. However, it is intended for displaying HTML document in a browser. Due to the flexibility of HTML syntax, the same web page can be implemented with completely different HTML codes. Since the DOM tree follows the HTML code exactly, it is not uncommon that two visually similar HTML documents have completely different DOM tree structures. This is generally undesirable when HTML documents are to be represented semantically.

For online journal articles, we observe that, as with scanned documents, the geometric layout is the most important cue for semantic organization. We, therefore, introduce another tree structure model, *zone tree*, which hierarchically groups the DOM nodes into zones based on the geometric layout of the web page.

Visually, a zone is a region on a web page that contains one or more DOM nodes. The root zone is the entire web page and includes only the <BODY> DOM node. The offspring DOM nodes of <BODY> are divided and then grouped into zones at different levels depending on their geometric relationships. The most important advantage of the zone tree model is that it is HTML tag independent. We believe that this zone tree model is better for organizing the related information within a document, and therefore better for information retrieval compared to the DOM tree model.

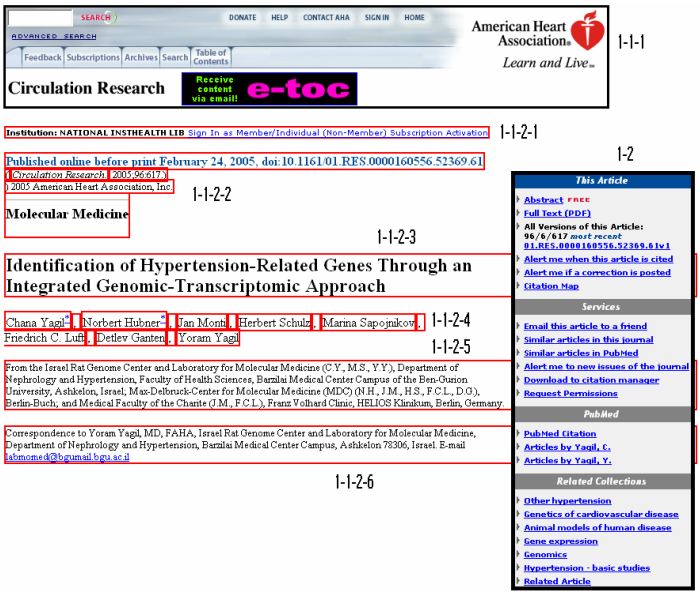


Figure 3. An example of zone tree structure.

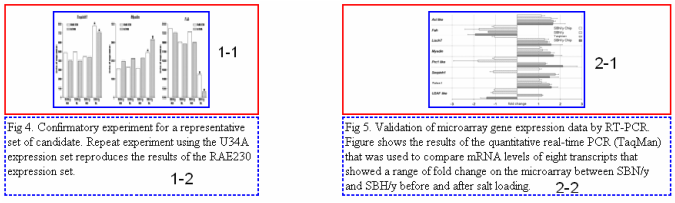
Figure 3 illustrates a portion of a real online article. Several zones are marked with bounding boxes of the inside DOM nodes and labeled with numbers.

The root zone, corresponding to <BODY> DOM node, is broken into Zone 1-1 and Zone 1-2. Zone 1-2 is on the right and includes a <TABLE> DOM node. Zone 1-1, which is not marked, includes the remaining components on the page. The gap between Zone 1-1-1 and Zone 1-1-2-1 is the largest and causes Zone 1-1 to be separated into Zone 1-1-1 and Zone 1-1-2 (not marked in the picture). The gaps between adjacent zones from Zone 1-1-2-1 to Zone 1-1-2-6 are the same. Zone 1-1-2 is therefore further divided into these 6 zones.

Zones 1-2 and 1-1-1 correspond to <TABLE> DOM nodes and have sub-trees under them. Zone 1-1-2-2 is also not a leaf zone, because it actually contains line break nodes, e.g.,
's. Although it has more than a dozen DOM nodes inside, Zone 1-1-2-4 is a leaf zone, because all the inside DOM nodes are inline nodes. Although depending on the application, one may want to analyze the inside DOM nodes to retrieve, for example, the name of each individual author, we decide that, for zone tree model, the zones containing only inline DOM nodes are leaf zones. Zones 1-1-2-1, 1-1-2-3, 1-1-2-5 and 1-1-2-6 are therefore considered leaf zones.

Zone 1-1-2-4 clearly demonstrates an advantage of zone tree over DOM tree. In HTML documents, it is not uncommon for a text line to be broken into several DOM nodes in order to implement some special features. In this example, a <NOBR> tag is added for each author name to avoid breaking the name into different lines. A <SUP> and an <A> tag are added for * characters to make them superscripts and establish web links, respectively. To make things worse, these DOM nodes can be at significantly different levels of the DOM tree. In this example, all the commas are direct children of <BODY> nodes, while the * characters are several levels deep in the DOM tree. DOM tree structure is good for Browser rendering and manipulating HTML documents, but obviously cumbersome for information retrieval. In zone tree representation, because all the texts inside Zone 1-1-2-4 are geometrically connected, they naturally belong to one zone.

Worth mentioning also is that the HTML code of Zone 1-2, which is a <TABLE> node, is between the HTML codes of the text lines of "Published online …" and "(Circulation Research …" (top 2 lines of Zone 1-1-2-2). Therefore, in the DOM tree, those two text lines are always separated by the <TABLE> node. For the zone tree model, they can be grouped together at a certain level because they are geometrically much closer. In our opinion, grouping these two text lines is also semantically sound.



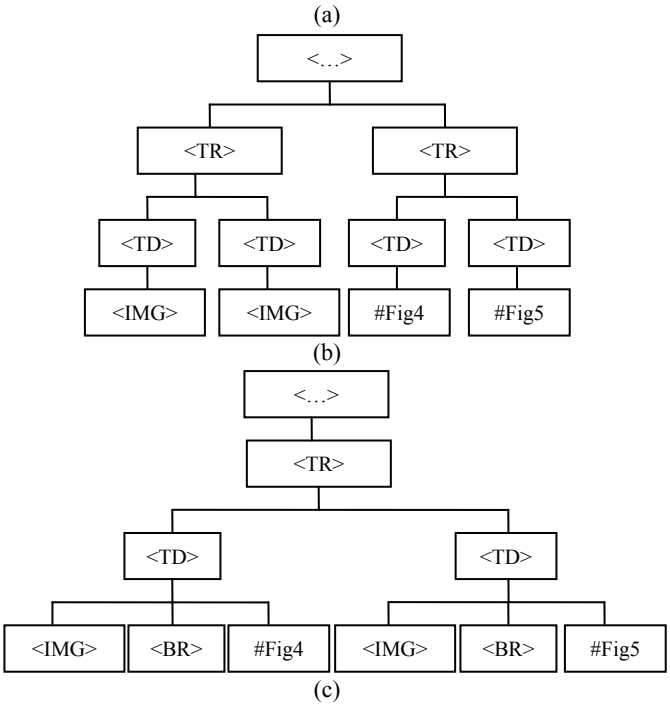


Figure 4. An HTML page showing two figures with their captions (a), which can be implemented with different HTML code, and therefore have different DOM trees, (b) and (c).

It is important to realize that the DOM and zone tree structures are generally different. DOM tree models the HTML syntax, while zone tree models the geometric layout of the HTML page. See another example in Figure 4. The page displays two figures with captions. It is easy to find several HTML implementations to realize the page. The DOM trees of two HTML implementations are also shown in the figure. They are quite different. In (b), the two figures are grouped together under a <TR> node and the two captions are grouped together under another <TR> node. In (c), each figure is grouped with its caption first.

Zone tree, on the other hand, is the same for both HTML implementations. As shown in Figure 4(a), Zone 1-1 and Zone 1-2 are grouped to form Zone 1, and Zone 2-1 and Zone 2-2 are grouped to form Zone 2, because they are geometrically close to each other.

To summarize, we describe the zone tree model as follows: The entire web page is the root zone node: $Z(D)=(N,S)$. $D=\{d_i\}$ is a set of DOM nodes inside this zone. The geometric position, i.e., upper-left and lower-right rectangular boundary coordinates, of zone Z is derived from these inside DOM nodes, i.e., finding the tightest bounding box of the DOM nodes. $N=\{Z_i\}$ is a set of children zones of zone Z. $S=\{s(Z_i,Z_j)\,|\,Z_i,Z_j\in Z\}$ is a set of separators separating children zones Z_i and Z_j. Recursively, child zone, $Z_i(D_i)=(N_i,S_i)$, has the same structure as Z. The leaf zone is the zone with only inline DOM nodes inside.

For a given HTML document, the zone tree generating process is to find a set of appropriate separators S to partition the Zone Z into a set of children zones N at each level of the tree. The process terminates at leaf zones. The algorithm detail is discussed in the next section.

Algorithm BuildZoneTree()
{
1. Render the HTML document on a browser. A DOM tree is then created by the browser.
2. Traverse the DOM tree to retrieve DOM node attributes, position information, and then label each DOM node as an insignificant, inline or line-break node.
3. Create a root zone of the whole page with <BODY> as the inside DOM node. Save it into a dividable zone list.
4. Choose a zone, pZone, from the dividable zone list, and do the following:
a. Analyze the immediate children of pZone's inside DOM nodes. If overlapping is detected, separate overlapping zones, save them into the dividable zone list, and go to step 5.
b. Analyze all offspring of pZone's inside DOM nodes, and collect leaf zones of pZone at this level. Leaf zones include:
zones corresponding to <TABLE> DOM nodes;
zones corresponding to the deepest line-break DOM nodes;
zones that include a set of consecutive inline DOM nodes.
c. Perform recursive X-Y cut on the collocation of the leaf zones to establish hierarchical tree structure.
d. Save zones with <TABLE> as inside DOM node into the dividable zone list. They need to be further analyzed.
5. Repeat 4 until dividable zone list is empty.
}

4. GENERATE ZONE TREE STRUCTURE

Given an HTML document, the zone tree is generated by combining DOM tree structure analysis and recursive X-Y cut. Given above is the high level process flow of the zone tree generation algorithm. Each step of the algorithm is explained in detail in the following subsections.

4.1 Retrieve attributes and position information of DOM nodes

We choose to render the HTML document on a WebBrowser ActiveX control of Microsoft Internet Explorer. It provides simple interfaces to create and access HTML DOM trees. During rendering, the DOM tree is created by the WebBrowser control. Performing a preorder traverse of the DOM tree, the attributes and position information of each DOM node can be easily retrieved through several interfaces.

The attributes extracted by the current algorithm include align, font-face, font-size, font-color and bgColor. The list can be expanded if necessary.

4.2 Label DOM nodes

We then label every DOM node to be either an insignificant, inline or line-break node, based on its tag and position information. This is accomplished with the following recursive function. Please note that it is a postorder traversal of the DOM tree, since we want to label a DOM node as a line-break node if at least one of its offspring is a line-break node.

```
Function LabelDomTree(pParent) //pParent is a DOM node
{
IF pParent is an insignificant node, label it as insignificant
IF pParent has no children OR pParent has only one text child
    Label it as either inline or line-break node based on its tag
FOR each child of pParent: pChild
    LabelDomTree (pChild)
IF all children of pParent are insignificant nodes, label pParent as insignificant
IF at least one of its children is a line-break node
    label pParent as a line-break node
ELSE
    label pParent as either inline node or line-break node depending on its tag
}
```

4.3 Generate zone tree

After the labeling process on the DOM tree is finished, we are ready to generate the zone tree. In the generation process, a dividable zone list is maintained. At the beginning, the root zone, corresponding to the whole page, is saved into the dividable zone list. Each zone in the dividable zone list is then retrieved and processed to create its sub-tree. The process continues until the dividable zone list is empty. The process on each dividable zone includes the following steps.

4.3.1 Separate overlapping DOM nodes

There can be overlapping children zones inside a zone. As shown in Figure 3, Zone 1-2, the right-aligned <TABLE>, overlaps with Zone 1-1-2-3, Zone 1-1-2-5 and Zone 1-1-2-6. The latter three encompass the whole width of the page. This overlap may seriously affect the subsequent recursive X-Y cut step. In our algorithm, once the overlapping zones are detected, they are immediately separated to form children zones. In the example, the page is divided into Zone 1-1 and Zone 1-2. The resulting children zones are saved into the dividable zone list.

Our experience shows that only <TABLE> nodes with the align attribute set as right, i.e., <TABLE align=right>, causes serious overlapping problems, which affect the following recursive X-Y cut step. Therefore, in the current algorithm, we restrict ourselves to separate only right-aligned <TABLE> zones.

4.3.2 Collect leaf zones inside a zone

If there are no overlapping children zones detected within a zone, we collect a set of leaf zones inside this zone at this level, such that the recursive X-Y cut can be applied.

There is no space between the consecutive inline nodes and they should be naturally merged together to form a leaf zone. <TABLE> DOM node is usually used by the author of HTML pages to group related information. We, therefore, choose to keep <TABLE> DOM node as a leaf zone at the level where it is found. <TABLE> zones are saved into the dividable zone list to be further divided at the next level. For line-break DOM nodes other than <TABLE>, we will always trace down until we reach the line break nodes, which contain only inline DOM nodes or have no children. Each of these deepest undividable line break DOM nodes also forms a leaf zone.

The following recursive function collects the leaf zones for a particular zone, pZone, where pDomNodes are the DOM nodes inside pZone.

```
Function CollectLeafZones (pDomNodes)
{
FOR each DOM node, pParent, of pDomNodes
{
IF pParent is a line-break node
{
Save the leaf zone formed by the previous inline DOM nodes.
IF pParent is a <TABLE> node
    Save pParent as a leaf zone.
ELSE IF pParent has no children or all children are inline DOM nodes
    Save pParent as a leaf zone
ELSE
    Get pParent's children, pChildren
    CollectLeafZones(pChildren)
}
ELSE
    Merge pParent with its previous adjacent inline or insignificant siblings
}}
```

4.3.3 Recursive X-Y cut on the merged zones

After the leaf zones are collected, the traditional recursive X-Y cut algorithm is applied to build the sub-zone-tree for the zone.

The next dividable zone is then retrieved from the dividable zone list and the leaf zone collection and recursive X-Y cut steps are repeated. The algorithm stops when dividable zone list is empty.

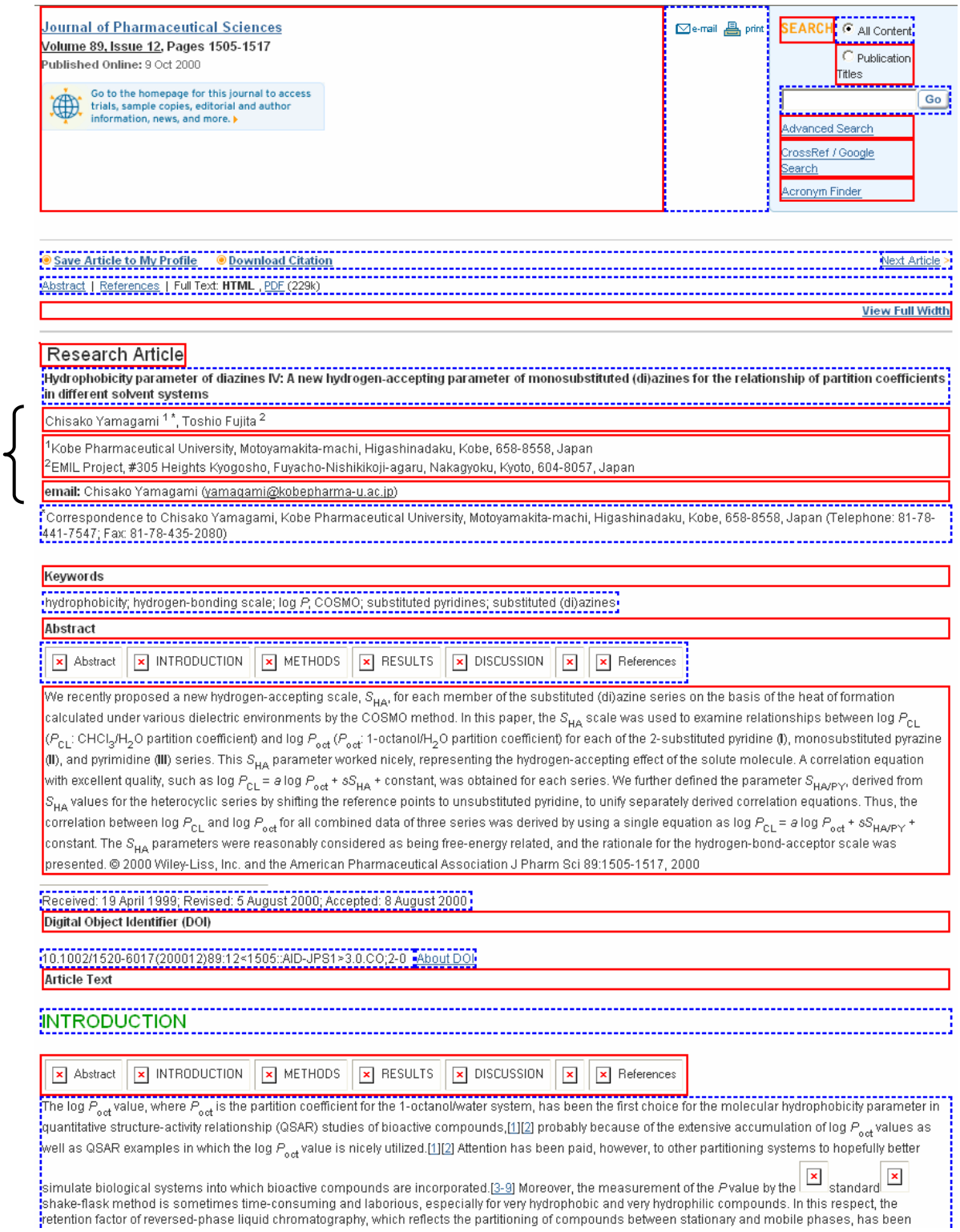
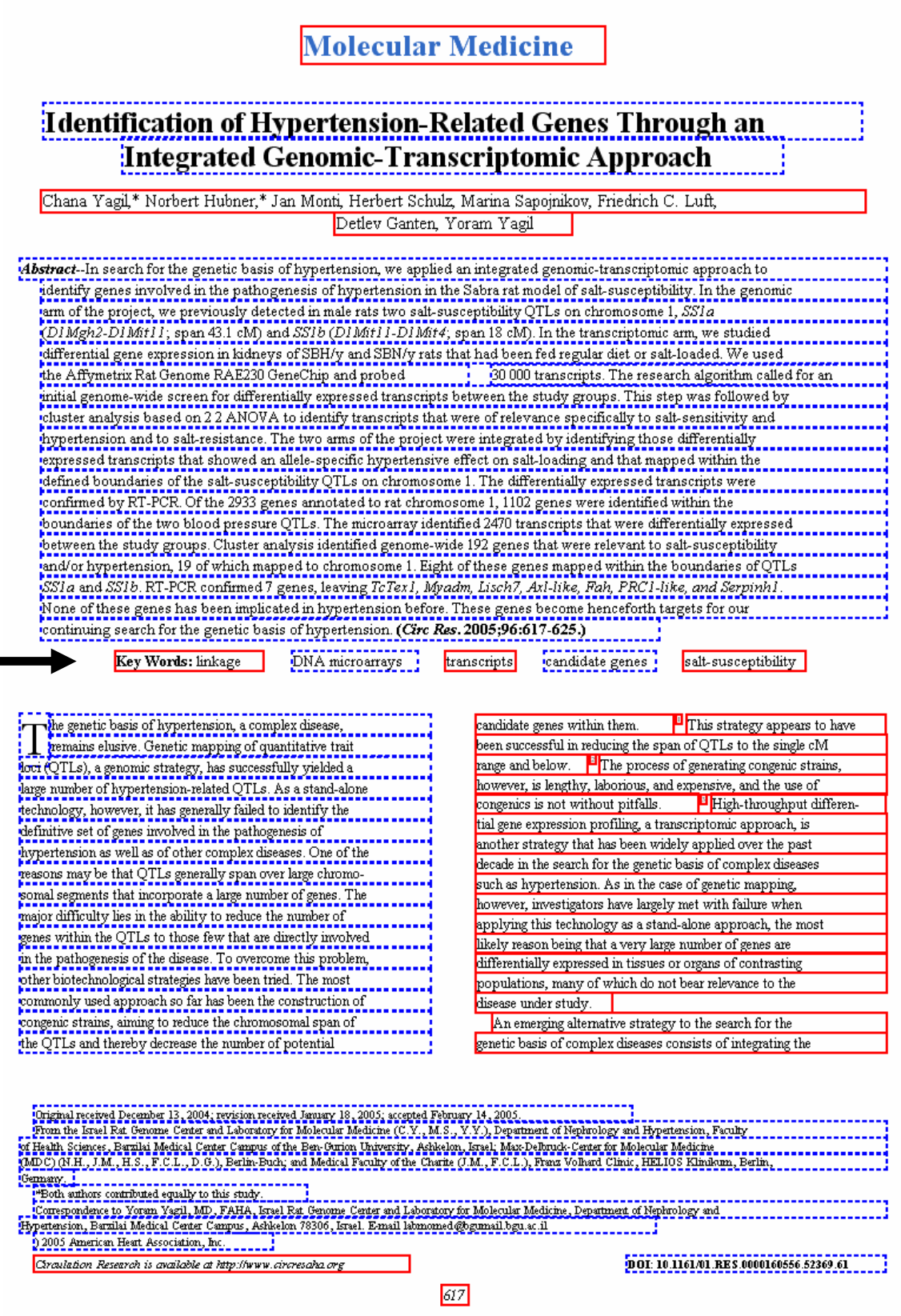


Figure 5. An example of segmentation results. Solid and dotted bounding boxes of inside DOM nodes alternate to indicate the result zones. The parenthesis indicates an under segmentation error. See text on error analysis for details.

Figure 6. An example of segmentation results. Solid and dotted bounding boxes of inside DOM nodes alternate to indicate the result zones.



Molecular Medicine

Identification of Hypertension-Related Genes Through an Integrated Genomic-Transcriptomic Approach

Chana Yagil,* Norbert Hubner,* Jan Monti, Herbert Schulz, Marina Sapojnikov, Friedrich C. Luft, Detlev Ganten, Yoram Yagil

Abstract--In search for the genetic basis of hypertension, we applied an integrated genomic-transcriptomic approach to identify genes involved in the pathogenesis of hypertension in the Sabra rat model of salt-susceptibility. In the genomic arm of the project, we previously detected in male rats two salt-susceptibility QTLs on chromosome 1, *SS1a (D1Mgh2-D1Mit11*; span 43.1 cM) and *SS1b (D1Mit11-D1Mit4*; span 18 cM). In the transcriptomic arm, we studied differential gene expression in kidneys of SBH/y and SBN/y rats that had been fed regular diet or salt-loaded. We used the Affymetrix Rat Genome RAE230 GeneChip and probed 30 000 transcripts. The research algorithm called for an initial genome-wide screen for differentially expressed transcripts between the study groups. This step was followed by cluster analysis based on 2 2 ANOVA to identify transcripts that were of relevance specifically to salt-sensitivity and hypertension and to salt-resistance. The two arms of the project were integrated by identifying those differentially expressed transcripts that showed an allele-specific hypertensive effect on salt-loading and that mapped within the defined boundaries of the salt-susceptibility QTLs on chromosome 1. The differentially expressed transcripts were confirmed by RT-PCR. Of the 2933 genes annotated to rat chromosome 1, 1102 genes were identified within the boundaries of the two blood pressure QTLs. The microarray identified 2470 transcripts that were differentially expressed between the study groups. Cluster analysis identified genome-wide 192 genes that were relevant to salt-susceptibility and/or hypertension, 19 of which mapped to chromosome 1. Eight of these genes mapped within the boundaries of QTLs *SS1a* and *SS1b*. RT-PCR confirmed 7 genes, leaving *TcTex1, Myadm, Lisch7, Axl-like, Fah, PRC1-like, and Serpinh1*. None of these genes has been implicated in hypertension before. These genes become henceforth targets for our continuing search for the genetic basis of hypertension. **(*Circ Res*. 2005;96:617-625.)**

Key Words: linkage DNA microarrays transcripts candidate genes salt-susceptibility

The genetic basis of hypertension, a complex disease, remains elusive. Genetic mapping of quantitative trait loci (QTLs), a genomic strategy, has successfully yielded a large number of hypertension-related QTLs. As a stand-alone technology, however, it has generally failed to identify the definitive set of genes involved in the pathogenesis of hypertension as well as of other complex diseases. One of the reasons may be that QTLs generally span over large chromosomal segments that incorporate a large number of genes. The major difficulty lies in the ability to reduce the number of genes within the QTLs to those few that are directly involved in the pathogenesis of the disease. To overcome this problem, other biotechnological strategies have been tried. The most commonly used approach so far has been the construction of congenic strains, aiming to reduce the chromosomal span of the QTLs and thereby decrease the number of potential candidate genes within them.[1] This strategy appears to have been successful in reducing the span of QTLs to the single cM range and below.[2] The process of generating congenic strains, however, is lengthy, laborious, and expensive, and the use of congenics is not without pitfalls.[3] High-throughput differential gene expression profiling, a transcriptomic approach, is another strategy that has been widely applied over the past decade in the search for the genetic basis of complex diseases such as hypertension. As in the case of genetic mapping, however, investigators have largely met with failure when applying this technology as a stand-alone approach, the most likely reason being that a very large number of genes are differentially expressed in tissues or organs of contrasting populations, many of which do not bear relevance to the disease under study.

An emerging alternative strategy to the search for the genetic basis of complex diseases consists of integrating the

Original received December 13, 2004; revision received January 18, 2005; accepted February 14, 2005.
From the Israel Rat Genome Center and Laboratory for Molecular Medicine (C.Y., M.S., Y.Y.), Department of Nephrology and Hypertension, Faculty of Health Sciences, Barzilai Medical Center Campus of the Ben-Gurion University, Ashkelon, Israel; Max-Delbruck-Center for Molecular Medicine (MDC) (N.H., J.M., H.S., F.C.L., D.G.), Berlin-Buch; and Medical Faculty of the Charite (J.M., F.C.L.), Franz Volhard Clinic, HELIOS Klinikum, Berlin, Germany.
*Both authors contributed equally to this study.
Correspondence to Yoram Yagil, MD, FAHA, Israel Rat Genome Center and Laboratory for Molecular Medicine, Department of Nephrology and Hypertension, Barzilai Medical Center Campus, Ashkelon 78306, Israel. E-mail labmomed@bgumail.bgu.ac.il

Circulation Research is available at http://www.circresaha.org
DOI: 10.1161/01.RES.0000160556.52369.61

Figure 7. An example of segmentation results. Solid and dotted bounding boxes of inside DOM nodes alternate to indicate the result zones. The arrow indicates an over segmentation error. See text on error analysis for details.

5. SEGMENT HTML DOCUMENT

The algorithm described in Section 4 generates a zone tree structure to represent the whole page by combining geometric layout and DOM tree analysis.

In a real application, the HTML document is required to be segmented into a set of blocks, not a tree structure. Therefore, a set of rules are required to prune the zone tree and select a set of zone nodes which appropriately segments the HTML document.

We regard this issue application dependent. Different applications require different specific rules. The ultimate goal of our system is to extract bibliographic data from online journal articles to help build a citation database, MEDLINE. Currently, we apply the following rules on each zone tree node in a preorder traversal of the zone tree to select a set of zone nodes. The HTML page is then segmented into blocks.

Rule 1: If the zone has only insignificant DOM nodes, stop.

Rule 2: If the zone is a leaf zone, stop and save this zone into the result.

Rule 3: If there is no text inside the zone or the text inside the zone is all space characters, stop and save this zone into the result. (In our application, we extract information from text only, and therefore, we stop when there is no text inside the zone.)

Rule 4: If the gap between two adjacent children zones is large enough, (empirically set to 6 pixels) we continue.

Rule 5: If the visual appearance of two zones is significantly different, we continue. The visual appearance is specified by the attributes of HTML tags, and can be easily retrieved through the interfaces provided by the WebBrowser control. In our experiments, we consider the following as significant changes in appearance: the background color (bgColor) is different, or the font attributes (size, color or face) are different.

6. EVALUATION

The experimental set consists of 104 articles from 11 journals. The manually-segmented zones of these 104 articles are considered as ground truth. There are 9726 zones, and the algorithm correctly identified 9376 zones, giving an accuracy of 96.40%.

No systematic evaluation on zone tree is conducted due to the lack of evaluation metric and the difficulty of creating ground truth. Careful subjective examination on more than 100 pages shows that most of the zone trees are near perfect and none is unacceptable.

Figures 5, 6 and 7 show three typical examples. The zones are indicated with solid red and dotted blue bounding boxes of inside DOM nodes. Please note that one zone may include several DOM nodes, and therefore several bounding boxes. We alternate the solid red and dotted blue bounding boxes to distinguish different zones. The first two are regular HTML files. They are from different publishers, and implemented in quite different styles. The third one is a PDF-converted-HTML file. The algorithm segments all three of them well.

Our error analysis shows that the major error types include:

- Some significant visual appearance changes are not important cues for semantic segmentation.

  This is the most common error. As shown in Figure 8, the ground truth segments the region into 3 zones, indicated with thick black bounding boxes and named as Zone 1-1, 1-2 and 1-3. The algorithm over-segments Zone 1-1 and Zone 1-3, indicated with solid red and dotted blue bounding boxes.

  The algorithm breaks Zone 1-1 into two zones due to the significant background color (bgColor attribute) change. (However, no error is counted for this one, since we compare only the text information of a zone, and the zone with red solid bounding box has the same text of the ground truth Zone 1-1.) The algorithm breaks Zone 1-3 into three zones due to the font color changes.

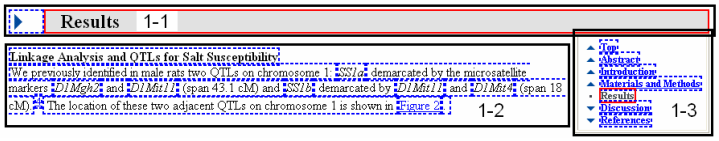


Figure 8. Errors due to significant visual appearance change.

- Geometry and visual appearance information are not enough to make the correct segmentation.

  As shown in Figure 5, indicated with a parenthesis. In the ground truth, this region is segmented into 3 zones, corresponding to author name, affiliation and Email address. Our algorithm keeps the region as one zone, because they have the same font attributes, the same background color, and the line spacing (gap) between them is small. It is difficult to separate them unless the meaning of the text is analyzed.

- In PDF converted HTML, sparse texts usually cause over segmentation.

  This is the most common error in a PDF-converted-HTML document. The PDF conversion library drops the figures and table rules. This leaves sparse texts, as shown in Figures 9. Our current algorithm usually oversegments them due to large gaps. Another example is indicated with a thick arrow in Figure 7. Because of large spaces, the algorithm breaks those key words.

Fortunately, in most of the above mentioned errors, the ground-truth zones have corresponding zone nodes in the zone tree representation. The error is either over segmentation or under segmentation. Therefore, it is possible for the subsequent information retrieval module to traverse the zone tree to find the correct zones: going down towards leaves in the zone tree if under segmentation happens; or going up towards the root in the zone tree if over segmentation occurs.

7. SUMMARY AND CONCLUSIONS

We have proposed a zone tree model to represent HTML documents of online medical journal articles. The model is primarily based on the geometric layout of the web page. We have also presented an algorithm to generate the zone tree and then segment the HTML documents by combining DOM tree analysis and recursive X-Y cut algorithm. Experimental results show that more than 95% zones can be correctly segmented.

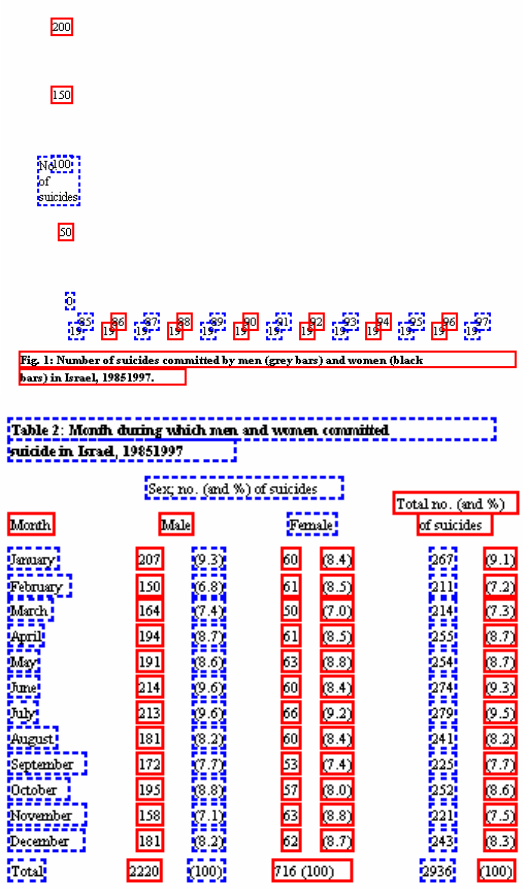


Table 2: Month during which men and women committed suicide in Israel, 19851997

Month	Sex; no. (and %) of suicides				Total no. (and %) of suicides	
	Male		Female			
January	207	(9.3)	60	(8.4)	267	(9.1)
February	150	(6.8)	61	(8.5)	211	(7.2)
March	164	(7.4)	50	(7.0)	214	(7.3)
April	194	(8.7)	61	(8.5)	255	(8.7)
May	191	(8.6)	63	(8.8)	254	(8.7)
June	214	(9.6)	60	(8.4)	274	(9.3)
July	213	(9.6)	66	(9.2)	279	(9.5)
August	181	(8.2)	60	(8.4)	241	(8.2)
September	172	(7.7)	53	(7.4)	225	(7.7)
October	195	(8.8)	57	(8.0)	252	(8.6)
November	158	(7.1)	63	(8.8)	221	(7.5)
December	181	(8.2)	62	(8.7)	243	(8.3)
Total	2220	(100)	716	(100)	2936	(100)

Figure 9. In a PDF-Converted-HTML file, the figures and the table rules are lost, which causes sparse texts. These sparse texts are handled inadequately by the current algorithm.

The most important property of the zone tree is that it is HTML tag independent. We consider the zone tree as a representation model of web pages at the semantic level. Useful operations may be defined on this tree model, such that the subsequent modules can retrieve and correlate information more efficiently and robustly.

The current zone tree model depends mostly on geometric layout of the web pages. We are investigating the incorporation of linguistic constraints, which are also HTML tag independent, to increase the accuracy and the reliability of the HTML page segmentation.

8. ACKNOWLEDGMENTS

This research was supported by the Intramural Research Program of the National Institutes of Health (NIH), National Library of Medicine (NLM), and Lister Hill National Center for Biomedical Communications (LHNCBC). We acknowledge useful discussions held with Deng Cai of the University of Illinois.

9. REFERENCES

[1] Baird, H.S., Jones, S.E., and Fortune, S.J., Image Segmentation by Shape-Directed Covers, *Proc. International Conference Pattern Recognition*, pp. 820-825, 1990.

[2] Buyukkokten, O., Garcia-Molina, H., and Paepche, A., Accordion Summary for End-Game Browsing on PDAs and Cellular Phones, *Proc. of Conference on Human Factors in Computer Systems*, 2001.

[3] Cai, D., Yu, S., Wen, J.-R., and Ma, W.-Y., Extracting Content Structure for Web Pages Based on Visual Representation, *Proc. of 5th Asia Pacific Web Conference*, 2003.

[4] Cai, D., Yu, S., Wen J.-R., and Ma, W.-Y., *VIPS: a Vision-Based Page Segmentation Algorithm*, Microsoft Technical Report (MSR-TR-2003-79), 2003.

[5] Chen, J., Zhou, B., Shi, J., Zhang, H., and Wu, Q., Function-Based Object Model towards Website Adaptation, *Proc. 10th International World Wide Web Conference*, 2001.

[6] Diao, Y., Lu, H., Chen, S., and Tian, Z., Toward Learning Based Web Query Processing, *Proc. of International Conference on Very Large Databases*, pp. 317-328, 2000.

[7] Ha, J., Haralick, R., and Phillips, I., Recursive X-Y Cut Using Bounding Boxes of Connected Components, *Proc. 3rd International Conference Document Analysis and Recognition*, pp. 952-955, 1995.

[8] Hauser, S.E., Le D.X., and Thoma G.R., Automated zone correction in bitmapped document images, *Proc. SPIE: Document Recognition and Retrieval VII*, *SPIE Vol. 3976,* San Jose, CA, pp. 248-258, 2000.

[9] Jain, A.K. and Yu B., Document Representation and Its Application to Page Decomposition, *IEEE Trans. Pattern Recognition and Machine Intelligence*, vol. 20, no. 3, pp. 294-308, 1998.

[10] Kaasinen, E., Aaltonen, M., Kolari, J., Melakoski, S., and Laakko, T., Two Approaches to Bringing Internet Services to WAP Devices, *Proc. 9th International World Wide Web Conference*, pp. 231-246, 2000.

[11] Lin, S.-H., and Ho, J.-M., Discovering Informative Content Blocks from Web Documents, *Proc. of ACM SIGKDD*, 2002.

[12] Marini, J., *The Document Object Model, Processing Structured Documents*, McGraw-Hill/Osborne, 2002.

[13] Nagy, G., Seth, S., and Viswanathan, M., A Prototype Document Image Analysis System for Technical Journals, *Computer*, vol. 25, pp. 10-22, 1992.

[14] Nagy, G., Twenty Years of Document Image Analysis in PAMI, *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 22, no. 1, pp.38 – 62, 2000.

[15] O'Gorman, L., The Document Spectrum for Page Layout Analysis, *IEEE Trans. Pattern Recognition and Machine Intelligence*, vol. 15, pp. 1162-1173, 1993.

[16] Pavlidis, T., and Zhou, J., Page Segmentation and Classification, *Graphical Models and Image Processing*, vol. 54, pp. 484-496, 1992.

[17] http://www.w3.org/DOM/

[18] http://pdftohtml.sourceforge.net/