Online medical journal article layout analysis

Jie Zou*, Daniel Le, George R. Thoma
Lister Hill National Center for Biomedical Communications, National Library of Medicine
8600 Rockville Pike, Bethesda, MD 20894

ABSTRACT

We describe a physical and logical layout analysis algorithm, which is applied to segment and label online medical journal articles (regular HTML and PDF-Converted-HTML files). For these articles, the geometric layout of the Web page is the most important cue for physical layout analysis. The key to physical layout analysis is then to render the HTML file in a Web browser, so that the visual information in zones (composed of one or a set of HTML DOM nodes), especially their relative position, can be utilized. The recursive X-Y cut algorithm is adopted to construct a hierarchical zone tree structure. In logical layout analysis, both geometric and linguistic features are used. The HTML documents are modeled by a Hidden Markov Model with 16 states, and the Viterbi algorithm is then used to find the optimal label sequence, concluding the logical layout analysis.

Keywords: Document Layout Analysis, Document Object Model (DOM), Web Information Retrieval, HTML Document Segmentation, HTML Document Labeling, Hidden Markov Model, Viterbi Algorithm

1. INTRODUCTION

Maintaining MEDLINE®, the world's preeminent bibliographic database of the biomedical journal literature, containing over 14 million citations, is one of the most important tasks at the National Library of Medicine (NLM). At the current rate, NLM may be indexing over a million articles annually within the next five years, double the level just a year ago. It is therefore important to have an efficient and reliable automatic system to extract bibliographic data for MEDLINE from online journal articles.

Most current Web information retrieval systems regard Web pages as the smallest indivisible units and simply search the information in the entire Web page linearly without trying to understand the structure of the page. In the narrow domain of online journal articles, a Web page usually consists of navigation panels, advertisements, banners, decorations, and the article itself. The article itself can also be logically divided into several information zones, such as title, author names, affiliations, acknowledgement, references and so on. We believe that, similar to the traditional scanned documents, physical and logical layout analysis of the online journal article HTML Web pages can expedite the subsequent information extraction processes and significantly increase their reliability.

Many existing Web information retrieval systems are designed for general Web pages. Without specific domain knowledge, HTML document segmentation (physical layout analysis) and labeling (logical layout analysis) are difficult, and therefore have not received much attention. In many such systems, HTML document segmentation is usually just a small component. Most of them simply use the HTML tags as indicators. For example, Diao et. al. used four types of tags, <P>, <TABLE>, <LI>/<UL> and <H1>~<H6> to detect four major types of segments: paragraphs, tables, lists and headings[4]. Lin and Ho used only <TABLE> tag to partition a page into several blocks[10]. Similarly, Buyukkokten et. al. and Kaasinen et. al. chose to use several simple tags, such as <P>, <TABLE> and <UL> to divide the Web page for subsequent conversion and summarization[1,7].

*jzou@mail.nlm.nih.gov; phone 1 301 496-7086; fax 1 301 402-0341;

VIPS[2,3] (VIsion-based Page Segmentation) renders the HTML document in a Web browser, so that some visual features, including spatial layout, background colors and so on, can be extracted and utilized in the segmentation. VIPS uses a tree structure to model a page. Each tree node corresponds to a block in a page, and has a value to indicate Degree of Coherence (DoC) [2,3]. The DOM (Document Object Model) tree is analyzed from root to leaves and the DOM nodes are separated or grouped by the visual features. This process continues until the DoC of the leaf tree node meets the pre-defined DoC. VIPS is designed to segment any kinds of Web pages. Due to the variations encountered in typical Web pages, a set of complicated heuristic rules is defined to calculate DoC and decide whether to divide a particular DOM node.

With the specific domain knowledge that we have, we are able to avoid making decisions based on complicated and incomparable features, and concentrate only on the dominant cues in the domain of interest. In our previous work[16], we showed that geometric relationships among DOM nodes are the most important cues for segmenting (physical layout analysis) online journal article Web pages, and have successfully designed a physical layout analysis algorithm to segment the whole Web page into zones. In addition, for online journal article Web pages, we can assign logical labels, such as title, authors, affiliations, and so on, to the zones.

As in the case of VIPS, we render the HTML document on a Web browser, thereby obtaining a visual image of the document. The physical and logical layout analyses of traditional scanned document images have been extensively studied and documented in the literature[14]. Borrowing from these well-studied document layout analysis algorithms and combining them with DOM tree analysis, we develop an approach for the physical and logical layout analysis (segmentation and labeling) of medical journal article HTML pages.

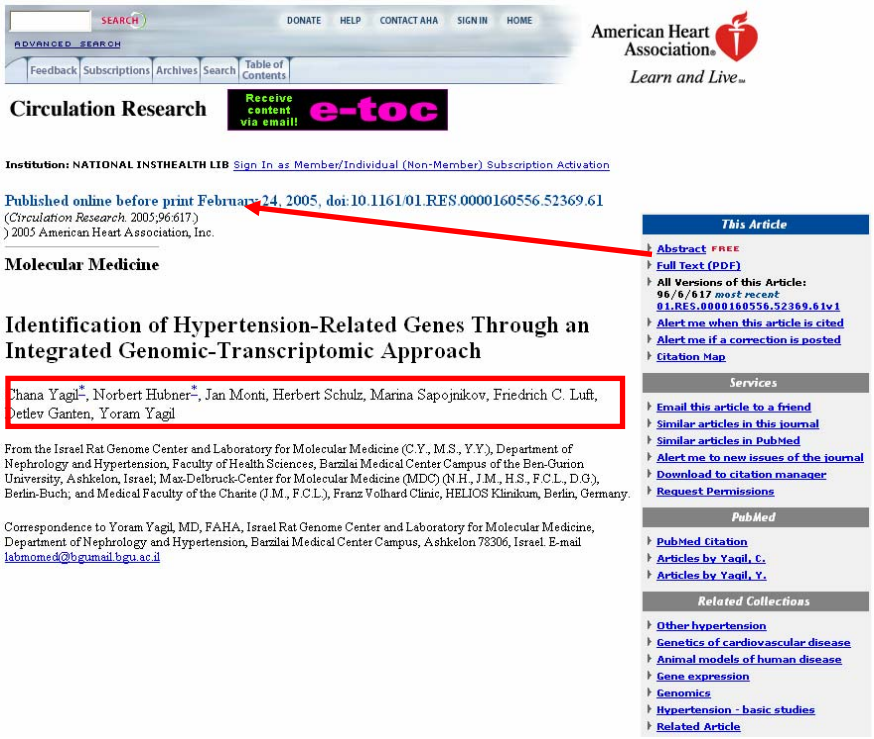
Document Object Model, the well-known model for HTML and XML documents, is published by the World Wide Web Consortium (W3C) and has been extensively applied in various applications, mostly for displaying and manipulating HTML documents. The drawbacks of using DOM for semantic understanding of HTML pages have been discussed in our previous work[16], but are briefly reviewed in Section 2. In Section 3, we describe our algorithm for physical layout analysis. It is an improved version of our previous algorithm[16]. In particular, the algorithm is simplified and adapted to the segmentation of the HTML document by automatically choosing values for certain parameters. In Section 4, we discuss our Hidden Markov Model (HMM) based HTML document logical layout analysis algorithm. Summary and conclusions constitute Section 5.

2. DOCUMENT OBJECT MODEL (DOM)

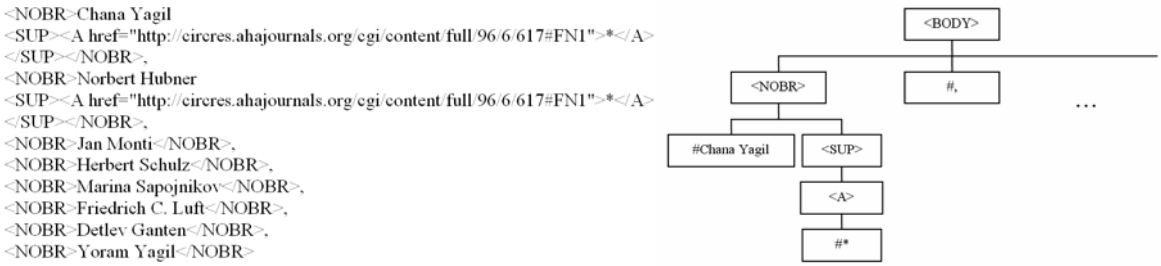
The Document Object Model (DOM) specification represents a significant advance in handling structured documents, including HTML and XML. In brief, DOM is a set of platform and language independent application programming interfaces (APIs) that describe how to access and manipulate information stored in HTML or XML documents[12,17].

Although DOM is a well defined document model, it is mostly for displaying and manipulating, but not for understanding HTML documents. In Figure 1, we point out a few drawbacks of using DOM to model HTML pages when semantic understanding of the Web page is the goal.

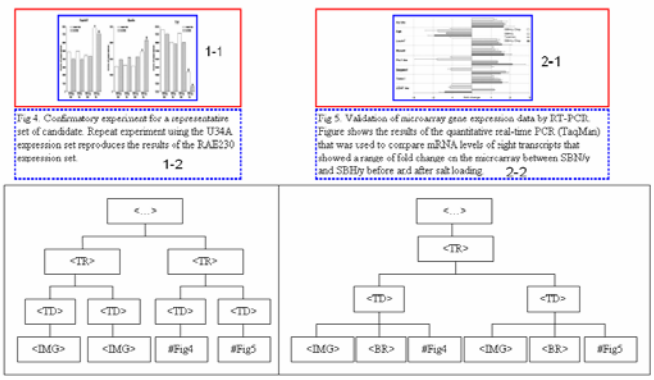
- **The DOM nodes may not be in a semantically meaningful order.** As shown in Figure 1(a) with an arrow, in the original HTML codes and therefore in the DOM tree, the navigation panel, corresponding to a <TABLE> node, is between the text lines of "Published online …" and "(Circulation Research …)". The reason probably is that the author of the HTML page wants the navigation panel to appear in that specific row. A much more semantically sound order would be to separate the navigation panel out and group the other text lines.
- **Simple text lines can be broken into several nodes at different levels of a DOM tree.** The actual HTML code snip and its corresponding DOM sub-tree of the author region (indicated with a bounding box) in Figure 1(a) are shown in Figure 1(b). In order to implement certain features, the simple text line is broken into a complicated DOM sub-tree. To make things worse, these DOM nodes can be at significantly different levels of the DOM tree. In the example, all the commas are direct children of <BODY> nodes, while the * characters are several levels deep in the DOM tree. DOM tree is obviously cumbersome for information retrieval.



(a)



(b)



(c)

Fig. 1. Drawbacks of using DOM directly for semantically understanding HTML documents.

- **Visually similar pages can have completely different DOM trees.** DOM tree models HTML syntax. Due to the flexibility of HTML syntax, visually similar Web pages can be implemented in different ways and therefore have completely different DOM trees. As in Figure 1(c), the top part is a simple Web page fragment displaying two figures with captions. It is easy to find several HTML implementations to realize the page. Two of them are shown in the bottom part of Figure 1(c). It is clear that these two DOM trees are completely different, obviously undesirable for semantic understanding of the Web page. Visually similar pages should have similar models.

Before leaving this section, we categorize the HTML DOM nodes into the following two types to clarify subsequent discussions.

Inline node: This type of node does not introduce line breaks. A complete list of inline node tags 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: This type of node does introduce line breaks. These include <TABLE>, <P>, <DIV> tags.

3. PHYSICAL LAYOUT ANALYSIS

Our physical layout analysis algorithm consists of the following four major steps.

1) Render the HTML document on a Web browser.
2) Generate a zone tree structure primarily based on the geometric relationships among DOM nodes.
3) Compute the statistics of the gaps (blank areas) between consecutive leaf zones.
4) Based on the statistics collected in step (3), prune the zone tree to generate the segmentation result.

We now discuss these steps in detail.

We choose to render the HTML document on a WebBrowser control of Microsoft Internet Explorer. The WebBrowser control provides simple interfaces to create and access HTML DOM trees. During rendering, the DOM tree is created by the WebBrowser control. Performing a preorder traversal of the DOM tree, the tag, text, attribute and position information of each DOM node can be easily retrieved through several interfaces (function calls). The DOM nodes are then labeled as either inline or line-break nodes.

There is no space between the consecutive inline nodes and they should be naturally merged together. We, therefore, traverse the DOM tree and merge the consecutive inline nodes into single zones. Merging consecutive inline nodes effectively prevents breaking single text lines, which would lead to over-segmentation. The zones formed by consecutive inline DOM nodes are named inline zones. On the other hand, every line-break DOM node forms a zone by itself, and we name them line-break zones. All inline zones and the line-break zones that correspond to leaf line-break DOM nodes constitute the complete set of leaf zones of the HTML Web page. Physical layout analysis basically organizes the leaf zones hierarchically according to their geometric relationship.

We choose to use a zone tree structure to represent the geometric relationships among the leaf zones. We notice that <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 mini page: the <TABLE> line-break zones are leaf zones at the level at which they are found; the same zone tree generation algorithm is applied on these mini pages to generate sub-zone-trees.

We adopt the classic recursive X-Y cut algorithm[13,6] to build zone trees due to its simplicity and efficiency. The algorithm recursively finds the largest horizontal or vertical gap among leaf zones and then partitions them at the largest gap. The major drawback of the X-Y cut algorithm is that it is sensitive to skew and noise. However, this is not a problem for online pages. The bounding boxes of DOM nodes are straight and clean.

Figure 2 illustrates an example of the zone tree structure. A journal article Web page is displayed in the left pane and the corresponding zone tree generated by the algorithm is in the right pane. BODY zone is the root of the zone tree and corresponds to the whole page. The page is then hierarchically divided into a zone tree structure. In the figure, Zone 29 is highlighted. The bounding boxes in the left pane indicate the components of Zone 29. It has six children. Zones 30,

40, 41, 42 and 43 are leaf zones, where Zones 30 (the first bounding box) and 41 (corresponding to the author region) are inline zones, and rest are line-break zones. The plus sign of Zone 31 (marked with a parenthesis) indicates that it has children because it contains several line-break DOM nodes. Worth mentioning is that the navigation panel (marked with an arrow) is separated out and the text lines are grouped into Zone 31 (See discussion of Figure 1(a)).

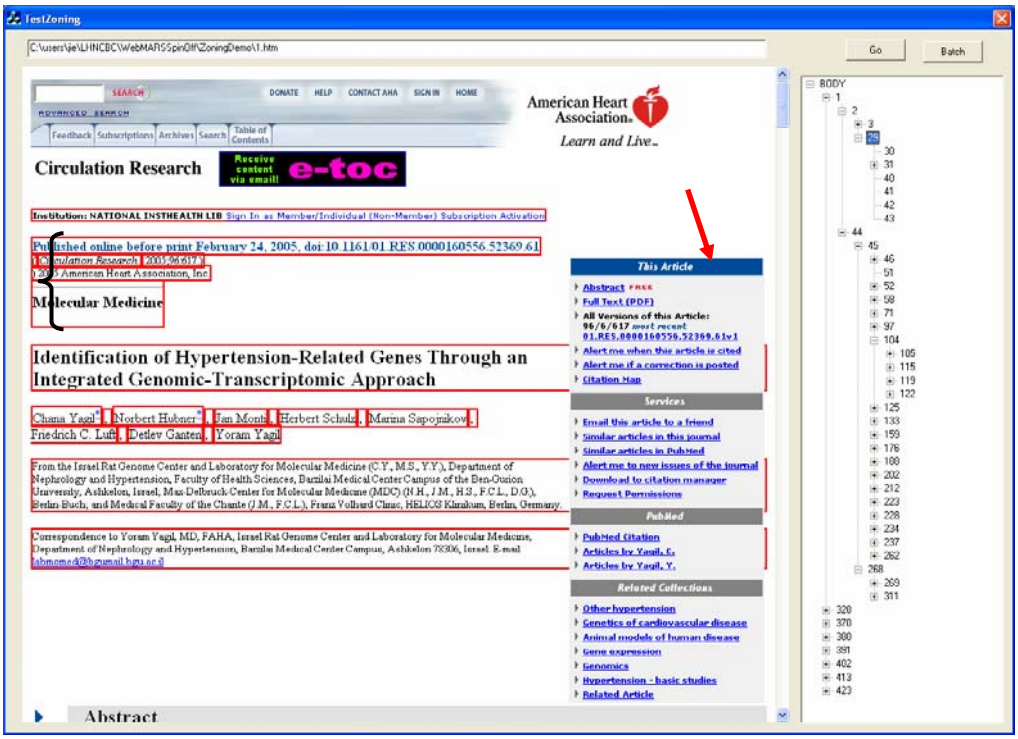


Fig. 2. An example of zone tree structure.

In real applications, the HTML document is required to be segmented into a set of blocks, not a tree structure. We therefore need to prune the zone tree to select a set of zone nodes which appropriately segment the HTML document. Since the zone tree is mostly generated according to the gaps between the zones, the key is then to find a gap threshold to decide where we should stop. These thresholds are adaptively selected for the HTML document being processed.

Many online journal articles are in PDF format. In order to standardize our input system and to minimize the number of modules for processing articles, we choose to convert PDF files into HTML files. In the PDF-Converted-HTML files, single text lines stand as single zones. For example, in a PDF-Converted-HTML file shown in the right part of Figure 3, the title breaks into two lines and the abstract constitutes 18 lines. In order to select a gap threshold adapted to the HTML page under processing, we collect all the gap sizes and build a histogram. The histogram is usually narrowly-peaked at the gap size, which corresponds to the line spacing. We can then easily detect the line spacing of the document by finding the most common gap, and the threshold is accordingly set to be just larger than the line spacing, so that the text lines are merged into a single zone.

In contrast to PDF-Converted-HTML files, paragraphs in regular HTML files usually stand as single zones (see the left part of Figure 3). We therefore select the gap threshold based on the paragraph spacing. We collect the gaps between consecutive leaf zones, which contain more than 20 words. In this case, the most common gap equals the spacing between paragraphs. We then accordingly set the threshold to be just smaller than the paragraph spacing, so that paragraphs are still stand-alone single zones.

After the gap threshold is selected, applying the threshold on each zone tree node in a preordered traversal of the zone tree generates the final segmentation result. Figure 3 shows the segmentation results for a regular HTML and a PDF-converted-HTML page. The segmentation results for both pages are good. However, in both pages, the authors and affiliations are grouped into one zone (indicated with parentheses) because gaps between them are small. Unless the text in the zones is analyzed, it is difficult to separate them based on geometric information only. Worth mentioning is that in

the zone tree structures, the zones do have children zones corresponding to the authors and affiliations. For the regular HTML (left), the zone has three children zones corresponding to authors, affiliations and email. For the PDF-converted-HTML (right), the zone has two children zones corresponding to authors and affiliations.

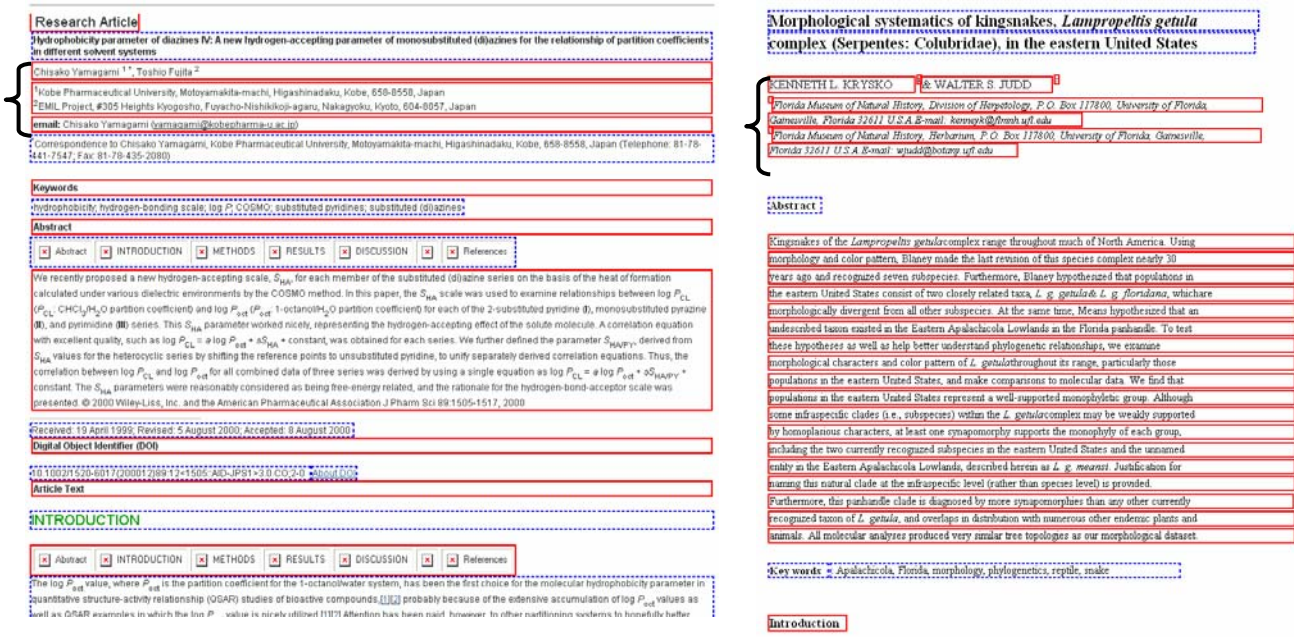


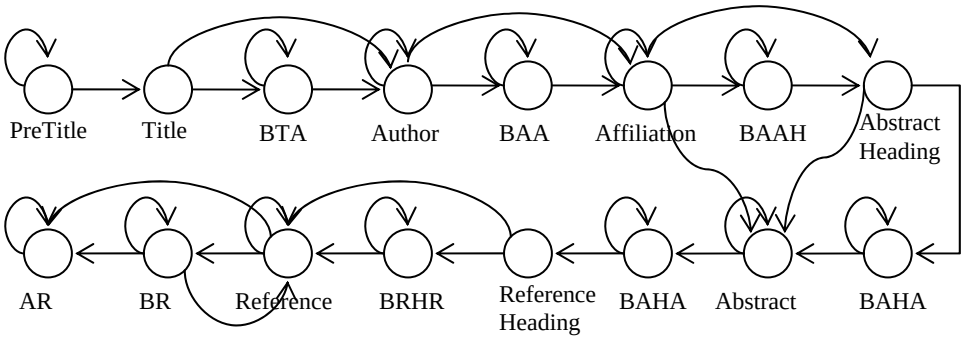
Fig. 3. Solid and dotted bounding boxes alternate to indicate the segmentation results of a regular HTML (left) and a PDF-Converted-HTML (right) page. For regular HTML pages, besides the gap threshold, other visual features, such as background color and font attributes (size, color and face), are also useful, and therefore utilized, for pruning the zone tree.

4. LOGICAL LAYOUT ANALYSIS

The logical components of an online journal article HTML page include title, author, affiliation, abstract, acknowledgement, references and so on. The goal of logical layout analysis is to detect and label these logical components of an HTML page. As shown in Figure 3, geometric information alone is insufficient for the logical layout analysis. Under-segmentation occurs in author and affiliation zones for both examples. Another important HTML-tag independent feature, text, remains to be analyzed.

Hidden Markov Model (HMM), introduced in the late 1960's, has been very successfully applied in speech recognition[15]. It also has been adopted in document image analysis, for extracting names and numbers from telephone yellow pages by Kopec and Chou[9], analyzing bilingual dictionaries by Kanungo and Mao[8], and preservation metadata extraction by Mao and Thoma[11].

We adopt HMM to model the logical components of the HTML journal articles. Our objective is to detect the following five logical components: *Title*, *Author*, *Affiliation*, *Abstract* and *References*. In the HTML journal articles, Abstract and References usually have headings, which are the zones containing informative words such as "Abstract" and "References". We therefore add two additional states, *Abstract Heading* and *Reference Heading*, into the model, such that these distinct landmarks can be utilized. In order to have a Barkis (left-right) like HMM, other zones are separated into 9 "other" states, depending on their positions relative to the above-mentioned 7 states. Our HMM therefore has a total of 16 states, and the complete HMM structure is illustrated in Figure 4.



BTA: Between Title and Author; *BAA*: Between Author and Affiliation; *BAAH*: Between Affiliation and Abstract Heading; *BAHA*: Between Abstract Heading and Abstract; *BRHR*: Between Reference Heading and References; *BR*: Between References; *AR*: After References.

Fig. 4. The HMM model for HTML journal articles. It is nearly a Barkis (left-right) model, except that Reference and BR (Between References) may iterate by themselves.

The state transition probability matrix, which is a 16 by 16 matrix, is estimated from 10 manually-labeled HTML journal articles. If there is no link between two states in Figure 4, the corresponding element in the state transition probability matrix is 0. In our algorithm, the initial state is always "PreTitle".

After the leaf zones are collected, as discussed in Section 3, the geometric and linguistic features are extracted. These include the normalized left and top coordinates of the zone, the height of the zone, individual words and the number of words in the zone.

We collected the word frequencies from 10 years of historic data from MEDLINE for title, authors, affiliations, and abstract. There are a total of 236,854 author names, and 432,545 distinct words. The twenty most frequent names (and name fragments) and their frequencies are shown in Table 1. There are 53 kinds of academic degrees that appear in the MEDLINE historic data (shown in Table 2). These are also helpful for detecting author zones. Some of the most frequent words in affiliations are listed in Table 3. Table 4 shows 12 possible headings for Abstracts of articles.

We do not eliminate stop words since they are useful for the classification. For example, the frequency of the word "in" in Title zones is about 63 times greater than that in Affiliation zones. MEDLINE citations do not contain Reference and "other" information. We merge the Title word and Author word collections to build the word frequency for Reference zones. For the other zones, we now simply use the word frequency in Abstract zones. These frequency statistics are to be updated when a large number of collections of the zones are available, possibly after experimenting with our HTML labeling algorithm for a while. On the other hand, as we will show, even with imprecise word frequency estimation for many states, due to the precise HMM document model, zones can still be accurately classified.

After the features (both geometric and linguistic) of all the leaf zones are extracted, the likelihoods of these leaf zones are calculated with the assumption that all the features are independent (Naïve Bayesian). Then, the well-known Viterbi algorithm[5,15] is applied to find the optimal component sequence, which concludes the labeling process.

Table 1. The 20 most frequent names collected from MEDLINE historic data.

van	de	Lee	Kim	Wang	Chen	Smith	Li	Liu	Zhang
0.68%	0.55%	0.35%	0.27%	0.27%	0.26%	0.24%	0.18%	0.16%	0.15%
Miller	Johnson	Suzuki	Tanaka	Williams	Brown	Lin	Yang	Martin	Jones
0.15%	0.14%	0.13%	0.13%	0.13%	0.13%	0.12%	0.12%	0.12%	0.12%

Table 2. 53 kinds of academic degrees appear in MEDLINE.

BA	BS	BOptom	BSc	DPhil	DSc	DA	DCh	DDS	DM
DMD	DPH	DPM	EdD	FC Path	FF Path	FIMLS	FRC Path	FRCS	FRCSI
FAAO	FACP	FRACP	FRCA	FRCAG	FRC Path	FRCR	FRCS	FRS	FRSC
MA	MB	MB ChB	MD	MHS	MRC Path	MS	MSc	MBA	MBBS
MDiv	MHS	MPH	M Phil	MRCOG	MRCP	MSW	OD	ORL	PhD
PharmD	RC Path	RN							

Table 3. 30 most frequent words in the Affiliation field in MEDLINE citations.

department	university	USA	school	hospital	center	Institute	research	Japan	college
Germany	division	national	UK	New York	France	laboratory	Italy	centre	Canada
California	clinical	state	ST	Netherlands	London	Australia	Boston	Texas	Sweden

Table 4. The possible headings for the Abstract field in MEDLINE.

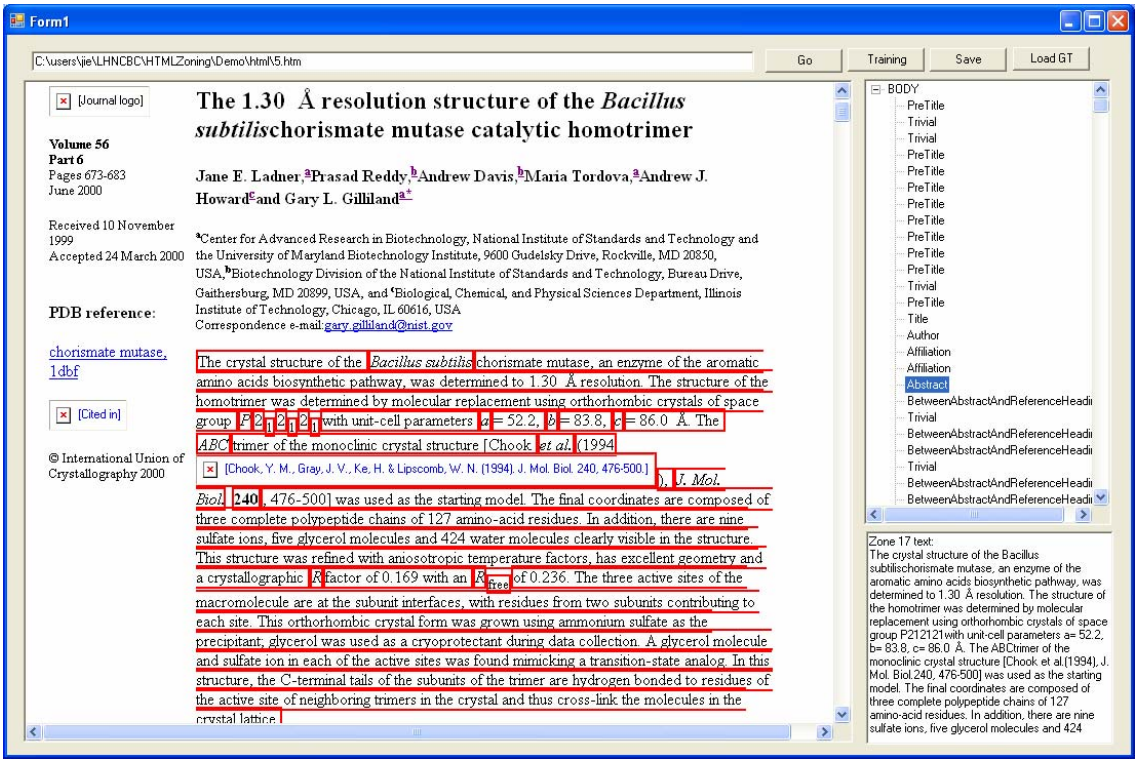
abstract	aim	background	contents	objective	objectives
presentation summary	purpose	study objective	study objectives	summary	synopsis

Figure 5 shows an example of labeling an HTML journal article. The left pane displays the HTML document, and the right pane is the label sequence inferred by the Hidden Markov Model and the Viterbi algorithm. When a label item in the right pane is selected, the corresponding zone is highlighted with bounding boxes, so that the labeling results can be visually examined. In Figure 5(a), the top portion of the article is displayed, with the abstract highlighted. Above the abstract, the title, author and affiliation zones are also correctly identified. Two affiliation zones correspond to the two paragraphs starting with “aCenter for …” and “Correspondence e-mail …” respectively. Note that the Abstract zone is correctly identified even though there is no Abstract Heading. We are interested in the text of the article only, therefore, the leaf zones which do not contain any non-blank texts, are labeled as Trivial zones. In Figure 5(b), the bottom portion of the article is displayed. Even though the references are in the concise format (the article title is missing and the journal title is significantly abbreviated), they are still correctly identified.

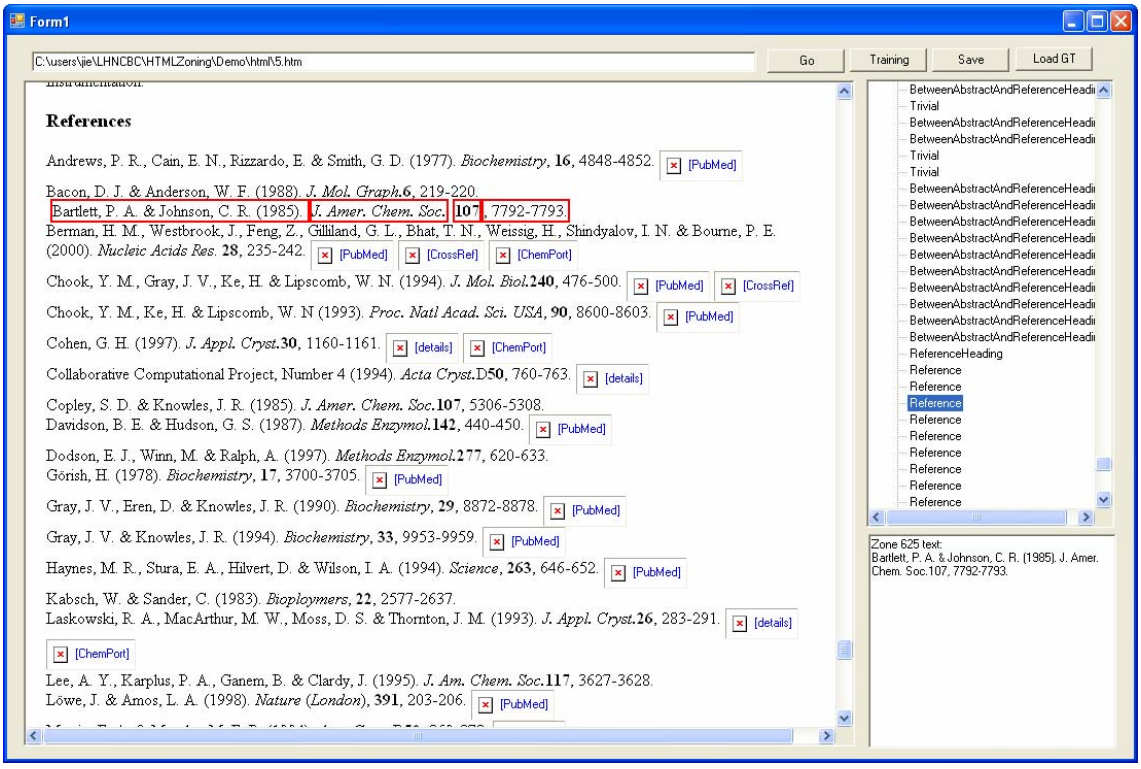
An important property of our HTML document logical layout analysis algorithm is that it minimizes the dependence on the HTML tags, and is therefore tolerant of different HTML implementation styles. Provided that the states (labels) do not switch their relative order in the document, the Hidden Markov Model and the labeling algorithm are applicable. Figure 6 shows the labeling results of two other HTML journal articles in completely different implementation styles, thereby demonstrating the versatility of the algorithm.

Preliminary evaluation was conducted with 15 HTML journal articles, all following different HTML implementation styles. Five of those are shown in Figures 2, 3 (left), 5, and 6. The labeling results are summarized in Table 5. The Title, Author, and Affiliation zones are all correctly identified, without any false positives. All Abstract and Reference zones are correctly identified also, but some false positives occur. Figure 7 illustrates reasons for the false positives. A larger scale evaluation (hundreds of HTML journal articles) is being conducted.

Occasionally, some journal articles, especially commentary and editorials may have their labels appearing in an atypical order. Appropriate Hidden Markov Models can be constructed easily for those cases by estimating the state transition probability matrix with a set of manually-labeled documents. It is expected that a small number of Hidden Markov Models will accommodate the vast majority of online journal articles.



(a)



(b)

Fig. 5. An example of logical labeling. (a) top portion; (b) bottom portion of an article.

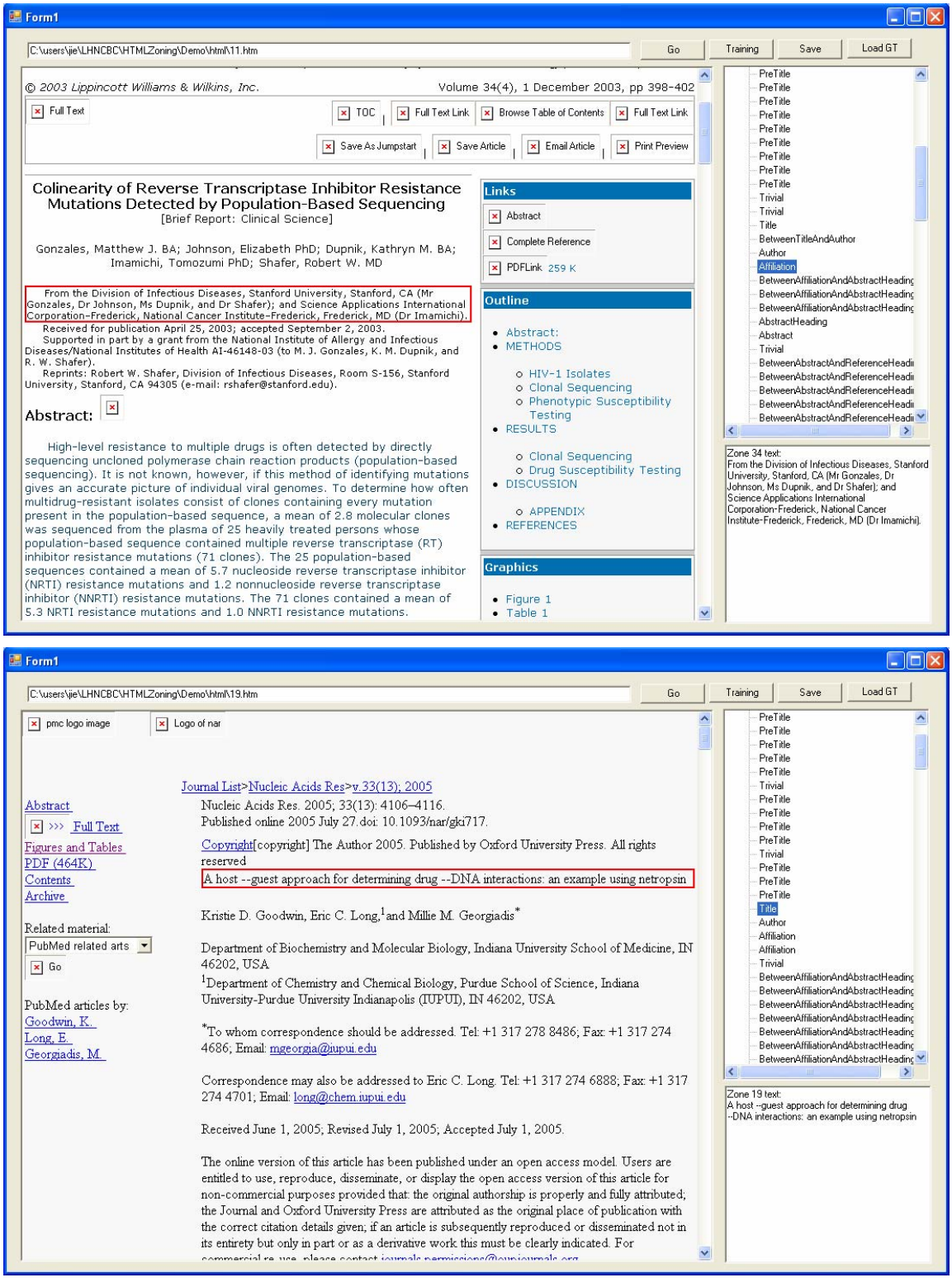
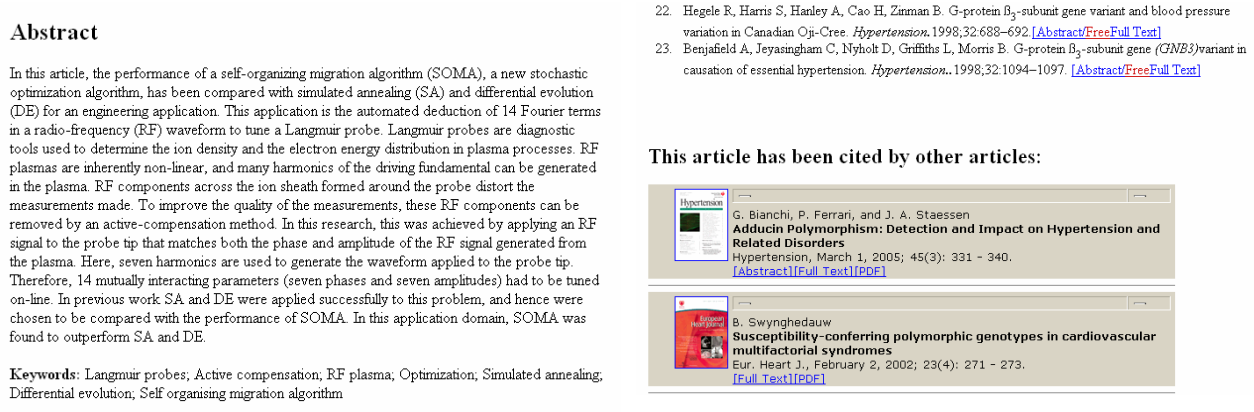


Fig. 6. Other examples of logical labeling.

Table 5. Logical labeling evaluation on 15 HTML journal articles in different implementation styles.

	Title	Author	Affiliation	Abstract	References
Total	15	15	25	30	593
Correctly Identified	15	15	25	30	593
False Positive	0	0	0	3	16



Abstract

In this article, the performance of a self-organizing migration algorithm (SOMA), a new stochastic optimization algorithm, has been compared with simulated annealing (SA) and differential evolution (DE) for an engineering application. This application is the automated deduction of 14 Fourier terms in a radio-frequency (RF) waveform to tune a Langmuir probe. Langmuir probes are diagnostic tools used to determine the ion density and the electron energy distribution in plasma processes. RF plasmas are inherently non-linear, and many harmonics of the driving fundamental can be generated in the plasma. RF components across the ion sheath formed around the probe distort the measurements made. To improve the quality of the measurements, these RF components can be removed by an active-compensation method. In this research, this was achieved by applying an RF signal to the probe tip that matches both the phase and amplitude of the RF signal generated from the plasma. Here, seven harmonics are used to generate the waveform applied to the probe tip. Therefore, 14 mutually interacting parameters (seven phases and seven amplitudes) had to be tuned on-line. In previous work SA and DE were applied successfully to this problem, and hence were chosen to be compared with the performance of SOMA. In this application domain, SOMA was found to outperform SA and DE.

Keywords: Langmuir probes; Active compensation; RF plasma; Optimization; Simulated annealing; Differential evolution; Self organising migration algorithm

22. Hegele R, Harris S, Hanley A, Cao H, Zinman B. G-protein β_3-subunit gene variant and blood pressure variation in Canadian Oji-Cree. *Hypertension*. 1998;32:688–692. [Abstract/FreeFull Text]
23. Benjafield A, Jeyasingham C, Nyholt D, Griffiths L, Morris B. G-protein β_3-subunit gene *(GNB3)*variant in causation of essential hypertension. *Hypertension*.. 1998;32:1094–1097. [Abstract/FreeFull Text]

This article has been cited by other articles:

G. Bianchi, P. Ferrari, and J. A. Staessen
Adducin Polymorphism: Detection and Impact on Hypertension and Related Disorders
Hypertension, March 1, 2005; 45(3): 331 - 340.
[Abstract][Full Text][PDF]

B. Swynghedauw
Susceptibility-conferring polymorphic genotypes in cardiovascular multifactorial syndromes
Eur. Heart J., February 2, 2002; 23(4): 271 - 273.
[Full Text][PDF]

Fig. 7. False positive examples. Left: the keywords zone is mislabeled as abstract. Right: the reverse references (cited by other articles) confuse the algorithm, and cause a few false positives.

5. CONCLUSION

We have described an HTML document layout analysis approach for segmenting and labeling online medical journal articles. The well-known Document Object Model (DOM) is designed for displaying and manipulating HTML documents, and is cumbersome for semantic level layout analysis. We have shown that geometric relationships among DOM nodes are the dominant cue for physical layout analysis and have designed an algorithm which can successfully generate zone tree structures to represent the spatial layout of the HTML pages, and then segment them into zones. In order to conduct logical layout analysis, i.e., assigning logical labels to the zones, both geometric and linguistic features are extracted. We adopt a Hidden Markov Model to constrain the label sequence of the HTML journal articles, and the Viterbi algorithm to find an optimal path for label sequence.

The important property of the proposed layout analysis algorithm is that it minimizes the dependence on HTML tags, and is therefore tolerant to variations in HTML implementation styles. Preliminary evaluation demonstrates that the algorithm performs successfully on HTML journal articles in various styles.

6. ACKNOWLEDGEMENT

We thank Dr. Jong Woo Kim for collecting the historic MEDLINE data and Dr. Song Mao for several valuable discussions. This research was supported by the Intramural Research Program of the National Institutes of Health (NIH), National Library of Medicine, and Lister Hill National Center for Biomedical Communications.

REFERENCES

1. O. Buyukkokten, H. Garcia-Molina and A. Paepche, "Accordion Summarization for End-Game Browsing on PDAs and Cellular Phones," *Proc. of the SIGCHI Conference on Human Factors in Computing Systems*, 213-220, (2001).
2. D. Cai, S. Yu, J.-R. Wen and W.-Y. Ma, "Extracting Content Structure for Web Pages Based on Visual Representation," *Proc. of 5th Asia Pacific Web Conference*, 406-417, (2003).
3. D. Cai, S. Yu, J.-R. Wen and W.-Y. Ma, *VIPS: a Vision-Based Page Segmentation Algorithm*, Microsoft Technical Report (MSR-TR-2003-79), 2003.
4. Y. Diao, H. Lu, S. Chen and Z. Tian, "Toward Learning Based Web Query Processing," *Proc. of International Conference on Very Large Databases*, 317-328 (2000).
5. G.D. Forney Jr., "The Viterbi Algorithm," *Proceedings of the IEEE*, 61(3), 268-278 (1973).
6. J. Ha, R. Haralick and I. Phillips, "Recursive X-Y Cut Using Bounding Boxes of Connected Components," *Proc. 3rd International Conference Document Analysis and Recognition*, 952-955 (1995).
7. E. Kaasinen, M. Aaltonen, J. Kolari, S. Melakoski and T. Laakko, "Two Approaches to Bringing Internet Services to WAP Devices," *Proc. 9th International World Wide Web Conference*, 231-246 (2000).
8. T. Kanungo and S. Mao, "Stochastic Language Models for Style-Directed Layout Analysis of Document Images," *IEEE Trans. Image Processing*, 12(5), 583-596 (2003).
9. G.E. Kopec and P.A. Chou, "Document Image Decoding Using Markov Source Models," *IEEE Trans. Pattern Recognition and Machine Intelligence*, 16(6), 602-617 (1994).
10. S.-H. Lin, and J.-M. Ho, "Discovering Informative Content Blocks from Web Documents," *Proc. of ACM SIGKDD*, 588-593, (2002).
11. S. Mao and G. Thoma, "Bayesian Learning of 2D Document Layout Models for Automated Preservation Metadata Extraction," *Proc. 4th IASTED International Conference on Visualization, Imaging, and Image Processing*, 329-334 (2004).
12. J. Marini, *The Document Object Model, Processing Structured Documents*, McGraw-Hill/Osborne, 2002.
13. G. Nagy, S. Seth and M. Viswanathan, "A Prototype Document Image Analysis System for Technical Journals," *Computer*, 25, 10-22 (1992).
14. G. Nagy, "Twenty Years of Document Image Analysis in PAMI," *IEEE Trans. Pattern Analysis and Machine Intelligence*, 22(1), 38 – 62 (2000).
15. L.R. Rabiner, "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition," *Proceedings of the IEEE*, 77(1), 257 – 286 (1989).
16. J. Zou, D. Le, G.R. Thoma, "Combining DOM tree and Geometric Layout Analysis for Online Medical Journal Article Segmentation," *Proc. Joint Conference on Digital Libraries*, 119-128 (2006).
17. http://www.w3.org/DOM/