

---

## Searching for web information more efficiently using presentational layout analysis

---

Miloš Kovačević\*

Faculty of Civil Engineering,  
University of Belgrade, Serbia, Yugoslavia  
Fax: +381-11-3370-223 E-mail: milos@grf.bg.ac.yu  
\*Corresponding author

Michelangelo Diligenti and Marco Gori

Dipartimento di Ingegneria dell'Informazione,  
University of Siena, Italy  
E-mail: diligmic@dii.unisi.it E-mail: marco@dii.unisi.it

Veljko Milutinović

Faculty of Electrical Engineering,  
University of Belgrade, Serbia, Yugoslavia  
E-mail: vm@etf.bg.ac.yu

**Abstract:** Extracting and processing information from web pages is an important task in many areas like constructing search engines, information retrieval, and data mining from the web. A common approach in the extraction process is to represent a page as a 'bag of words' and then to perform additional processing on such a flat representation. In this paper we propose a new, hierarchical representation that includes browser screen coordinates for every HTML object on a page. Using visual information one is able to define heuristics for recognition of common page areas such as a header, left and right menu, footer and the centre of a page. Initial experiments have shown that, using our heuristics, defined areas are recognised properly in 73% of cases. Finally, we introduce a classification system which, taking into account the proposed document layout analysis clearly outperforms standard systems by 10% or more.

**Keywords:** Web page classification; focused search engines; page layout; common areas; recognition heuristics; Naïve Bayes classifier.

**Reference** to this paper should be made as follows: Kovačević, M., Diligenti, M., Gori, M. and Milutinović, V. (2003) 'Searching for web information more efficiently using presentational layout analysis', *Int. J. Electronic Business*, Vol. 1, No. 3, pp.310-326.

**Biographical notes:** Miloš Kovačević received his BS and MS degrees in electrical engineering from the Faculty of Electrical Engineering, University of Belgrade, Yugoslavia in 1995 and 2001 respectively. He is currently a PhD student in the computer science department at the Faculty of Electrical Engineering, University of Belgrade. He works as a system engineer at the Faculty of Civil Engineering, University of Belgrade and his research interests are machine learning and computer networks.

Michelangelo Diligenti received his MS degree in telecommunication engineering from the University of Siena, Italy in 1998. He is currently a PhD student in informatics engineering at the University of Siena. His current research interests are applications of machine learning in web search technologies.

Marco Gori received his BS degree in electrical engineering from the University of Florence in 1984. He received his PhD degree in computer science from the University of Bologna, Italy in 1990. He is a full professor of computer science at the Faculty of Engineering, University of Siena, Italy. His current research interests are applications of machine learning in web search technologies. Professor Gori is a fellow member of IEEE.

Veljko Milutinović received his BS degree, MS degree and PhD degree in electrical engineering from the Faculty of Electrical Engineering, University of Belgrade, Yugoslavia in 1975, 1978 and 1982 respectively. He was a coarchitect of the first 32-bit 200 MHz RISC microprocessor (RCA 1986) He is a full professor of computer science at the Faculty of Electrical Engineering, Belgrade. His current research interests are e-business technologies and developing infrastructure architectures for e-business.

Paper presented to the International Conference on Data Mining ICDM2002, Maebachi City, Japan, December 2002.

---

## **1 Introduction**

Web pages are designed for humans! Whilst the previous sentence is more than obvious, still many machine learning and information retrieval techniques for processing web pages do not utilise implicit visual information contained in an HTML source. By visual information we mean the position of HTML objects in a browser window. For example, one can say that a certain image is in the top left corner of the screen or that the most informative paragraph is in the centre of the page and it occupies an area of 100x200 pixels.

Where can this kind of information be useful? Consider the problem of feature selection in document (web page) classification. There are several methods to perform the feature selection process such as Information Gain [1] or TF-IDF (term frequency – inverse document frequency) [2]. In both cases we try to estimate what are the most relevant words that describe document D i.e. the best vector representation of D that will be used in the classification process. Assuming that web pages are designed for their visual sense, we can argue that some words represent noise with respect to page topic if they belong to menu, banner link or perhaps page footer. That noise can be misleading for classifiers. Also, we can suppose that words that belong to the central part of the page (screen) carry more information than words from the bottom right corner. Hence, there should be a way to weigh differently words from different layout contexts. At the present moment, in classic algorithms, the positions of words and their spanning areas are not considered at all!

Let us mention another problem – designing an efficient crawling strategy for focused search engines. Given a specific topic T and a starting set of pages S, it is necessary to find as many T on-topic pages as possible in a predefined number of steps. By step is

meant visiting (and downloading and indexing) a page reachable from  $S$  following hyperlinks from pages in  $S$ . In other words it is important to estimate whether an outgoing link is promising or not. Different techniques have been described by others [3–5]. In any case when crawler decides to take into account a page for link expansion, all links from the page are inserted into the crawl frontier (links that are to be visited). Sometimes, links that belong to menus or footers can be misleading or less important than links from the centre of a page. Can we measure the importance of a link according to the link position in a page (on a browser screen)? Also, we can calculate link density in some areas of a page (screen) and weigh them taking into account that density factor. Links that are surrounded by ‘more’ text are probably more important to the topic than links positioned in groups, but groups of links can signify we are on the hub page that can also be important to our focused crawler. Can we learn the positions of interesting links for some topics? In any case, we believe, information about position and belonging to a certain area can help to infer whether the link is promising or not!

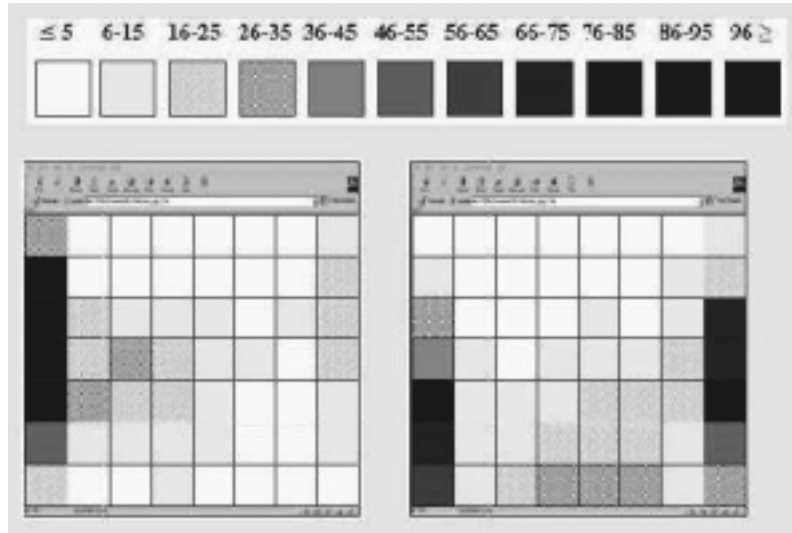
To note the final example, consider the problem of cheating search engines by inserting irrelevant keywords into an HTML source. This is a widely used technique in order to raise the probability of indexing a page by search engine and representing it with a higher rank among search results. Whilst it is relatively easy to detect and reject false keywords where their foreground colour is the same as the background colour, there is no way to detect keywords of regular colour but covered with images. If the coordinates of objects in a page representation are known, then search engines could filter false keywords hidden by other objects and users would get better answers to their queries!

All the previously mentioned issues motivated us to define a new representation of a page extracted from an HTML source, which includes visual information, and to show how it can be utilised in the recognition of common areas in a web page. However, we were additionally encouraged to do this work when discovering the fact that users expect web designers to put certain objects in predefined areas on the browser screen [6]. Figure 1 shows where users expect to find internal and external links. In [6] one can find other examples of user expectation about the positions of common page objects. As we shall see, the previous fact and the corresponding real situation on the web will help us to define heuristics for recognition of common page areas (menus, header, footer and ‘centre’ of a page).

In order to show the strength of the visual representation of a page we went one step further. We defined a possible feature selection method that partitions a page into recognised areas and produces related subsets of features processed by different Naïve Bayes classifiers. It is shown that differently weighted class probability estimates, taken all together in a linear combination, lead us to more accurate classification systems. The classification system presented in this paper is still evolving and currently outperforms common by 10%.

In the ongoing research we plan to construct a more efficient focused crawling system based on presentational layout analysis. This focused system should provide a cost-effective means for companies to acquire needed information, making them more competitive in the global marketplace.

**Figure 1** User expectation in percentages concerning the positions of internal (left) and external (right) links in the browser window [6]. It is clear that menus are supposed to be either inside the left or the right margin of a page



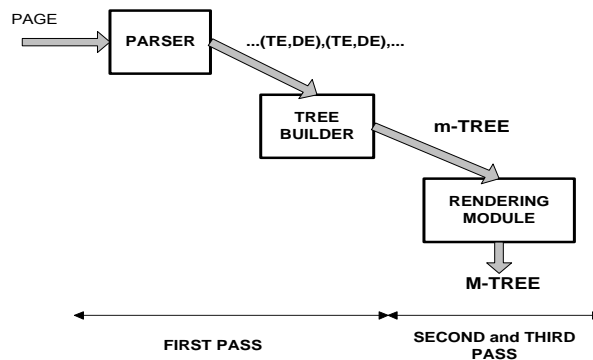
The outline of the paper is as follows: In Section 2 we define the *M-Tree* format of a page used to render the page on the virtual screen, i.e. to obtain coordinates for every HTML object. Section 3 describes heuristics for recognition of a header, footer, left and right menu, and a 'centre' of a page. In Section 4 experimental results in a recognition process on a predefined dataset are shown. In Section 5, we present a page classification system that utilises previously extracted visual information. Also, experimental results in a classification process on a predefined dataset are shown. In Section 6 some business implications of our research are explained. The main hypothesis is that the web represents one of the most important sources of business data. Layout analysis leads us to more efficient technology solutions needed for the construction of an intelligent web acquisition system capable of collecting that information. Finally, conclusions are given in Section 7.

## 2 Extracting visual information from an HTML source

We introduce a virtual screen (VS) that defines a coordinate system for specifying the positions of HTML objects inside pages. The VS is a half strip with a predefined width and an infinite height, both measured in pixels. The VS is set to correspond to a page display area in a maximised browser window on a standard monitor with a resolution of 1024x768 pixels. Of course one can set any desirable resolution. The width of the VS is set to 1000 because when vertical scroll bars from browser are removed, that quantity is usually left for rendering the page. Obviously, pages are of different lengths and so theoretically height can be infinite. The top left corner of the VS represents the origin of the VS coordinate system.

Now, the process of visual information extraction will be described. The main operations applied in the extraction process are shown in Figure 2. In the first step a page is parsed using an HTML parser that extracts two different types of elements – tags and data. Tag elements (TE) are delimited with  $\langle \rangle$  while data elements (DE) are contained between two consecutive tags. Each TE includes the name of the corresponding tag and a list of attribute-value pairs. DE is represented as a list of tokens, which taken all together form the data string between consecutive tags. Separators between tokens are white space characters and are omitted from the DE list of tokens. The DE contains an empty list if no token is placed between two consecutive tags (i.e. input stream is  $\dots \rangle W \langle \dots$ ,  $W$ -white space character or none). The parser skips  $\langle \text{SCRIPT} \rangle$  and  $\langle !-\dots \rangle$  tags.

**Figure 2** Constructing the *M-Tree* (main steps)



In the second step, as soon as the  $\langle \text{TE,DE} \rangle$  pair is extracted from the input HTML stream, it is injected into the tree builder. Tree builder applies stack machine and a set of predefined rules to build a tree that represents the HTML structure of the page. The output of this component we named *m-Tree*. There are many papers that describe the construction of the parsing tree of an HTML page [7,8]. In our approach a technique is adopted which constructs the tree in one single pass through the given page, i.e. parsing and building the *m-Tree* is done in a single pass. Rules are used to properly nest TEs into the hierarchy according to the HTML 4.01 specification [9]. Additional efforts were made to design a tree builder that will be immune on bad HTML source. Now *m-Tree* (in further text *mT*) will be defined.

*Definition 1: mT is directed n-ary tree defined with a set of nodes N and a set of edges E with following characteristics:*

1  $N = N_{\text{desc}} \cup N_{\text{cont}} \cup N_{\text{data}}$  where:

-  $N_{\text{desc}}$  (description nodes) is a set of nodes, which correspond to TEs of the following HTML tags:  $\{\langle \text{TITLE} \rangle, \langle \text{META} \rangle\}$

-  $N_{\text{cont}}$  (container nodes) is a set of nodes, which correspond to TEs of the following HTML tags:  $\{\langle \text{TABLE} \rangle, \langle \text{CAPTION} \rangle, \langle \text{TH} \rangle, \langle \text{TD} \rangle, \langle \text{TR} \rangle, \langle \text{P} \rangle, \langle \text{CENTER} \rangle, \langle \text{DIV} \rangle, \langle \text{BLOCKQUOTE} \rangle, \langle \text{ADDRESS} \rangle, \langle \text{PRE} \rangle, \langle \text{H1} \rangle, \langle \text{H2} \rangle, \langle \text{H3} \rangle, \langle \text{H4} \rangle, \langle \text{H5} \rangle, \langle \text{H6} \rangle, \langle \text{OL} \rangle, \langle \text{UL} \rangle, \langle \text{LI} \rangle, \langle \text{MENU} \rangle, \langle \text{DIR} \rangle, \langle \text{DL} \rangle, \langle \text{DT} \rangle, \langle \text{DD} \rangle, \langle \text{A} \rangle, \langle \text{IMG} \rangle, \langle \text{BR} \rangle, \langle \text{HR} \rangle\}$

-  $N_{\text{data}}$  (data nodes) is a set of nodes, which correspond to the DEs.

Each  $n \in N$  has the following attributes: *name* equals the name of the corresponding tag except for  $N_{data}$  nodes where *name* = "TEXT", *attval* is a list of attribute-value pairs extracted from the corresponding tag and can be null (i.e. nodes from  $N_{data}$  have this attribute set to null). Additionally, each  $N_{data}$  node has four more attributes: *value*, *fsize*, *emph*, and *align*. The first contains tokens from the corresponding DE, the second describes the font size of these tokens, third carries information as to whether the tokens belong to the scope of validity of one or more of the following HTML tags: {<B>, <I>, <U>, <STRONG>, <EM>, <SMALL>, <BIG>}. The last one describes the alignment of the text (left, right or centred). In further text, if  $n$  corresponds to tag  $X$  we write  $n_{<X>}$  ( $n$  has *name* =  $X$ ).

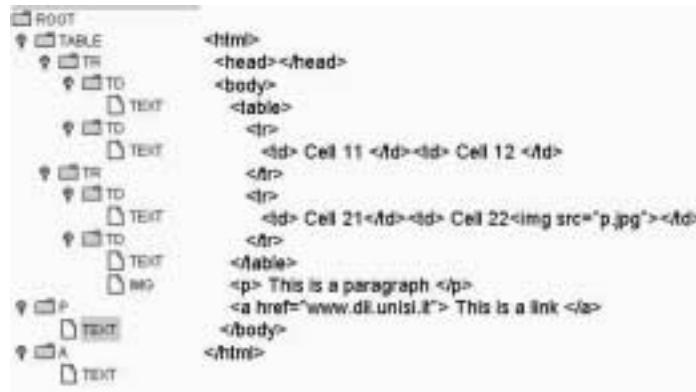
- 2 Root of  $mT$ ,  $n_{ROOT} \in N_{cont}$  represents a page as a whole and its *name* is set to "ROOT" while *attval* contains only one pair (URL : source url of the page itself)
- 3  $E = \{(n_x, n_y) \mid n_x, n_y \in N\}$ .

There can be only the following types of edges:

- $(n_{ROOT}, n_{desc}), n_{desc} \in N_{desc}$
- $(n_{cont1}, n_{cont2}), n_{cont1} \in N_{cont} \setminus \{n_{<IMG>}\}, n_{cont2} \in N_{cont} \setminus \{n_{ROOT}\}$  if  $n_{cont2}$  belongs to the context of  $n_{cont1}$  according to the nesting rules of the HTML 4.01 specification
- $(n_{cont}, n_{data}), n_{cont} \in N_{cont} \setminus \{n_{<IMG>}\}, n_{data} \in N_{data}$  if  $n_{data}$  belongs to the context of  $n_{cont}$

From definition 1 it is clear that the image and text nodes can only be leafs in an  $mT$ . Figure 3 shows a possible example of a simple page and its corresponding  $mT$ .

**Figure 3** An HTML source (right) and a related  $mT$  (left)



After the  $mT$  is obtained from the input page and when the context of every object of interest is known, it is possible to apply the algorithm for coordinate calculation. In fact, it is nearly the same algorithm that every browser does when rendering the page. Coordinates of objects are calculated in the third step using the rendering module (see Figure 2) and constructed  $mT$  as its input. We did not find any specific algorithm for

page rendering except some recommendations from W3C [9] and so it was necessary to design our own. We decided to imitate the visual behaviour of the internet Explorer because of the popularity of this product. It is clear how difficult it could be if all aspects of the rendering process are to be taken into account. Hence some simplifications are made, which in our opinion do not influence significantly the final task – the recognition of common areas in a page. The simplifications are as follows:

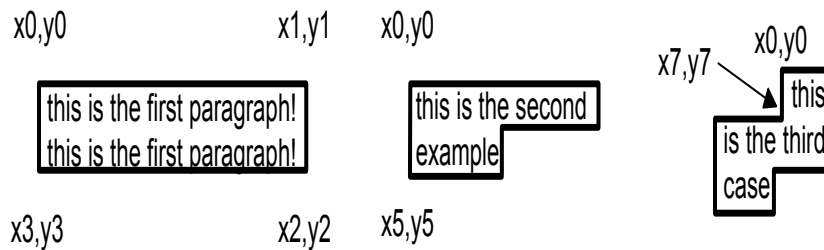
- 1 Rendering module (RM) calculates only coordinates for nodes in  $mT$ , i.e. HTML tags out of definition 1 are skipped.
- 2 RM does not support layered HTML documents.
- 3 RM does not support frames.
- 4 RM does not support style sheets.

The rendering module produces the final, desirable representation of a page – *M-Tree* (in further text *MT*). *MT* extends the concept of *mT* by incorporating coordinates for each  $n \in N \setminus N_{desc}$ .

*Definition 2: MT is the extension of mT with the following characteristics:*

- 1 For  $\forall n \in N \setminus N_{desc}$ , there are two additional attributes: *X* and *Y*. These are arrays which contain *x* and *y* coordinates of the corresponding object polygon on the VS.
- 2 If  $n \in N_{cont} \setminus \{n_{<A>}\}$  then *X* and *Y* have dimension 4 and it is assumed that the object represented by *n* occupies rectangle area on the VS. The margins of this *n*'s rectangle are:
  - The down margin is equal to the up margin of the left neighbour node if it exists. If *n* does not have a left neighbour or  $n = n_{<TD>}$  then the down margin is equal to the down margin of *n*'s immediate parent. If  $n = n_{ROOT}$  then the down margin is the *x*-axis of the VS coordinate system.
  - The up margin is equal to the up margin of the rightmost leaf node in the subtree in which *n* is the root node.
  - The left margin is equal to the left margin of *n*'s immediate parent, shifted to the right for the correction factor. This factor depends on the name of the node (i.e. if *name* = LI this factor is set to 5 times the current font width because of the indentation of list items). If  $n = n_{<TD>}$  and *n* has a left neighbour then the left margin is equal to the right margin of *n*'s left neighbour. If  $n = n_{ROOT}$  then the left margin is the *y*-axis of the VS coordinate system.
  - The right margin is equal to the right margin of *n*'s immediate parent. If  $n = n_{<TABLE>}$  or  $n = n_{<TD>}$  then the right margin is set to correspond to table/cell width.
- 3 If  $n \in N_{data}$  or  $n = n_{<A>}$  then *X* and *Y* can have dimension from 4 to 8 depending on the area on the VS occupied by the corresponding text/link (see Figure 4). Coordinates are calculated assuming the number of characters contained in the value attribute and current font width. Text flow is restricted to the right margin of the parent node and then new line is started. Heights of lines are determined by current font height.

**Figure 4** Some of the possible *TEXT* polygons



The previous definition covers most aspects of the rendering process but not all, because of the complexity of the process. For example if the page contains tables then RM implements the modified auto-layout algorithm [9] for calculating table/column/cell widths. So when  $n_{\langle \text{TABLE} \rangle}$  is encountered, RM makes one more pass from that node down the  $mT$  to calculate the cell/column/table widths. Hence, the first pass is dedicated to table width calculations, and in the second pass RM calculates final coordinates for the nodes that belong to the observed subtree. If there are other  $n_{\langle \text{TABLE} \rangle}$  nodes down on the path (nesting of tables in the page) the process of calculating widths is recursively performed, but with no additional passes. Before resolving a table, artificial cells (nodes) are inserted in order to simplify the calculus in cases where cell spanning is present (colspan and rowspan attributes in a  $\langle \text{TD} \rangle$ ).

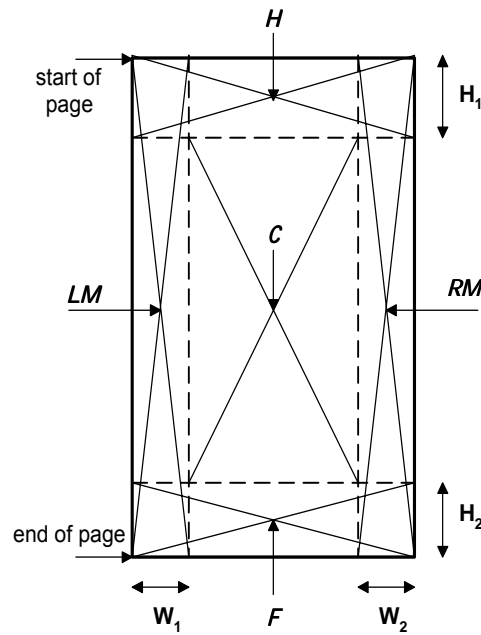
Let us consider the complexity of the  $MT$  extraction process. The first and second step (extracting  $\langle \text{TE}, \text{DE} \rangle$  pairs and building the  $mT$ ) are performed in a single pass through the page. Hence the complexity so far is  $O(s)$ , where  $s$  represents the size of the file. In the third step RM transforms  $mT$  into  $MT$  while passing through  $mT$  and calculating coordinates for every non-descriptive node. If  $mT$  does not contain nodes that represent table TEs (tables in a page) then one single pass in the third step is needed and the complexity remains linear. If the page contains tables then, in the worst case, RM performs one pass more. Hence the complexity of the RM phase is  $O(2s)$  and the resulting complexity of the  $MT$  extraction process is  $O(3s)$ , which is satisfactory for most applications.

### 3 Defining heuristics for recognition of common areas of interest – layout analysis

Given the  $MT$  of a page and assuming common web design patterns, it is possible to define a set of heuristics for recognition of standard areas in a page such as menu or footer. Firstly, the areas of interest are listed to be header (H), footer (F), left menu (LM), right menu (RM), and the centre of the page (C). At present there are no exact definitions in the open literature for these page areas (one can think of these areas as groups of objects). Therefore, we adopted intuitive definitions of these areas, which rely exclusively upon VS coordinates of logical groups of objects in a page. It is helpful to understand these groups of objects as frequently found areas in pages regardless of a page

topic. They are tightly related to the presentational concept of a page. Naturally, heuristics based on visual information are used to recognise them, instead of exact algorithms. After careful examination of many different pages on the web, we restricted the areas in which  $H$ ,  $F$ ,  $LM$ ,  $RM$ , and  $C$  can be found. Before we describe what is recognised to be  $H$ ,  $F$ ,  $LM$ ,  $RM$ , and  $C$ , we will introduce the specific partition of a page, as shown in Figure 5.

**Figure 5** Position of areas of interest in a page



We set  $W_1 = W_2$  to be 30% of the page width in pixels determined by the rightmost margin among the nodes from  $MT$ .  $W_1$  and  $W_2$  define  $LM$  and  $RM$  respectively, which are locations where  $LM$  and  $RM$  can be exclusively found. We set  $H_1 = 200$  pixels and  $H_2 = 150$  pixels.  $H_1$  and  $H_2$  define  $H$  and  $F$  respectively, which are locations where  $H$  and  $F$  can be exclusively found. Of course one can set different values, but initial experiments showed previous values are appropriate (see Section 4). Now we define the following heuristics:

*Heuristic 1:  $H$  consists of all nodes from  $MT$  that satisfy one or more of the following conditions:*

- 1 Subtree  $S$  of  $MT$  with its root  $r_S$  belongs to  $H$  if  $r_S$  is of type  $n_{\langle TABLE \rangle}$  and  $n_{\langle TABLE \rangle}$  completely belongs to  $H$  (i.e. the upper bound of a table is less than or equal to  $H_1$ ).
- 2 Subtree  $S$  of  $MT$  with its root  $r_S$  belongs to  $H$  if upper bound of  $r_S$  is less than or equal to  $m$  and  $r_S$  does not belong to the subtrees found in 1. Number  $m$  is the maximum upper bound of all  $n_{\langle TABLE \rangle}$  nodes found in 1.

*Heuristic 2: LM consists of all nodes from MT that are not contained in H and satisfy one or more of the following conditions:*

- 1 Subtree S of MT with its root  $r_S$  belongs to LM if  $r_S$  is of type  $n_{\langle \text{TABLE} \rangle}$  and  $n_{\langle \text{TABLE} \rangle}$  completely belongs to LM (i.e. the right bound of a table is less than or equal to  $W_1$ ).
- 2 Subtree S of MT with its root  $r_S$  belongs to LM if  $r_S$  is of type  $n_{\langle \text{TD} \rangle}$ , and  $n_{\langle \text{TD} \rangle}$  completely belongs to LM, and  $n_{\langle \text{TABLE} \rangle}$  to which this  $n_{\langle \text{TD} \rangle}$  belongs has lower bound less than or equal to  $H_1$ , and upper bound is greater than or equal to  $H_2$ .

*Heuristic 3: RM consists of all nodes from MT that are not contained in H, LM and satisfy one or more of the following conditions:*

(Similar to heuristic 2 except RM and  $W_2$  instead of LM and  $W_1$ )

*Heuristic 4: F consists of all nodes from MT that are not contained in H, LM, RM, and satisfy one or more of the following conditions:*

- 1 Subtree S of MT with its root  $r_S$  belongs to F if  $r_S$  is of type  $n_{\langle \text{TABLE} \rangle}$  and  $n_{\langle \text{TABLE} \rangle}$  completely belongs to F (i.e. the down bound of a table is greater than or equal to  $H_2$ ).
- 2 Subtree S of MT with its root  $r_S$  belongs to F if lower bound of  $r_S$  is greater than or equal to  $m$  and  $r_S$  does not belong to the subtrees found in 1. Number  $m$  is the maximum lower bound of all  $n_{\langle \text{TABLE} \rangle}$  nodes found in 1.
- 3 Let  $n \in \{n_{\langle \text{BR} \rangle}, n_{\langle \text{HR} \rangle}\}$  or  $n$  is in the scope of the central text alignment. Further, assume  $n$  is the lowest of all nodes in MT completely contained in F. Subtree S of MT with its root  $r_S$  belongs to F if lower bound of  $r_S$  is greater than or equal to upper bound of  $n$ , and  $r_S$  does not belong to the subtrees found in 1 and 2.

*Heuristic 5: C consists of all nodes from MT that are not in H, LM, RM, and F:*

From previous definitions of heuristics one can realise the importance of the  $\langle \text{TABLE} \rangle$  tag and its related tags  $\langle \text{TR} \rangle$  and  $\langle \text{TD} \rangle$ . These tags are commonly used ( $\gg 88\%$ ) for purposes not originally intended by the inventors of HTML [10]. Web designers usually organise the layout of the page and alignment of objects by including a lot of tables in a page. Therefore, every table cell often represents the smallest amount of logically grouped information, visually presented to the user in a browser window (in our case on the VS). The same stands for tables that often group menu objects, footers, search and input forms, and other common page objects. Realisation of the previous heuristics is done in, at most, 2 additional passes through the given MT. Hence the resulting complexity of the whole recognition process is nearly  $O(5s)$ , and that allows us to apply it in the different applications mentioned in Section 1.

#### 4 Experimental results in a recognition process

An experiment is performed to show how efficient the recognition process can be using only visual information given through *MT*. The set-up of the experiment was as follows:

- Step 1:* Construct the dataset *D* that contains a sufficient number of different pages from different sites.
- Step 2:* Walk through *D* manually and label areas that can be considered as H, F, LM, RM, and C.
- Step 3:* Perform automatic extraction of *MT* for each page in *D*. Perform automatic recognition of areas of interest using defined heuristics on *MT*. Make automatically labelled new dataset  $D_1$  from each previously processed *MT*.
- Step 4:* Walk through  $D_1$  manually and estimate how well areas are recognised using manually labelled *D* as a reference point.

Step 1 is conducted by downloading nearly 16000 pages from the open source directory [www.dmoz.org](http://www.dmoz.org) as a starting point for our crawler. We downloaded nearly 1000 files from the first level of each root category. *D* is constructed from the downloaded set by randomly choosing 515 files, uniformly distributed among categories and also in size. The main motivation for constructing the dataset of our own will be described in Section 5 where classification issues are covered. Two persons performed step 2 once. The second person was a kind of control and ultimate judge for labelling. Step 3 is performed using our *Siena Tree tool* [11] that includes an *MT* builder and logic for applying recognition heuristics. Again, two persons in step 4 make a judgment of recogniser performance by entering into each labelled file and comparing automatic labels with hand made labels from step 2. After step 4 we achieved the results shown in Table 1.

**Table 1** Success in a recognition process (in %)

	<i>Header</i>	<i>Footer</i>	<i>Left M</i>	<i>Right M</i>	<i>Overall</i>
Not recognised	25	13	6	5	3
Bad	16	17	15	14	24
Good	10	15	3	2	50
Excellent	49	55	76	79	23

Shaded rows represent successful recognition

In order to discuss the results from Table 1, the notions of ‘bad’ or ‘good’ in the recognition process have to be clarified. If area *X* exists but it is not labelled at all, or if *X* does not exist but something is labelled as *X*, then a mark ‘not recognised’ is evidenced. If less than 50% of objects that belong to *X* are labelled, or if some objects out of *X* are labelled too, then a mark ‘bad’ is evidenced. A mark ‘good’ is evidenced if more than 50% but less than 90% of objects from *X* are labelled and no objects out of *X* are labelled. A mark ‘excellent’ is evidenced if more than 90% of objects from *X* and no objects out of *X* are labelled. The verification process was very tedious and it lasted one week! We are also very aware that estimation of previously mentioned percentages is a subjective category due to a lack of strict definitions for such objects (areas).

We stress that a mark ‘bad’ is given in cases where something is wrongly recognised. That is because we intend to use *Siena Tree* to filter the noise for text classification purposes. Therefore, if some text from the centre of the page is wrongly removed we could lose important information. Also, recognition of *C* is, according to heuristic 5, complementary to the recognition of other areas. So we did not include it in the performance measurements. The results from the column ‘overall’ are obtained by introducing the total score *S* for the page *P* as a sum of all marks for recognition of all areas of interest. If  $X \in \{H, F, LM, RM\}$  is ‘not recognised’ then the corresponding mark is 0. Marks ‘bad’, ‘good’, and ‘excellent’ are mapped into 1, 2, and 3 respectively. Now, if  $S=12$  we assume the recognition process for the particular file (page) is ‘excellent’. Similarly ‘good’ stands for  $8 \leq S < 12$ , ‘bad’ stands for  $4 \leq S < 8$ , and ‘not recognised’ stands for  $S < 4$ . Analysing pages that perform as ‘bad’ or ‘not recognised’ we found that in nearly 20%, the *MT* was not quite correct but the *mT* was correct i.e. the rendering process was not good enough. A typical error is that portions of a page are internally good rendered but they are scrambled as a whole. For the rest of the 80% of ‘not recognised’ and ‘bad’ recognised pages we suppose the defined heuristics are not sufficient enough. Finally we selected values for margins  $H_1$ ,  $H_2$ ,  $W_1$ , and  $W_2$  according to statistics from [6]. In further research other values have to be considered as well.

## 5 Page classification using visual information

The rendering module provides an enhanced document representation, which can be used whenever the traditional bag-of-words representation cannot capture the complex structure of a web page (i.e. page ranking, crawling, document clustering and classification). In particular, we have performed some document classification experiments using a rich representation provided by the rendering module (*MT* representation). At the time of writing, there is not a dataset of web pages which has been commonly accepted as a standard reference for classification tasks. Thus, we have decided to create our own. After extracting all the URLs provided by the first 5 levels of the DMOZ topic taxonomy, we selected 14 topics at the first level of the hierarchy (we rejected topic ‘Regional’ which features many non-English documents). Each URL has been associated to the class (topic) from which it has been discovered. Finally, all classes have been randomly pruned, keeping only 1000 URLs for each class. Using a web crawler, we downloaded all the documents associated to the URLs. Many links were broken (server down or pages not available anymore), thus only about 10.000 pages could be effectively retrieved (an average of 668 pages for each class). These pages have been used to create the dataset. Such a dataset [12] can be easily replicated, enlarged and updated (the continuous changing of web format and styles does not allow the employment of a frozen dataset since, after a few months, it would not reflect the real situation that can be found on the internet).

### 5.1 Naive Bayes classifier

The Naive Bayes classifier [13] is the simplest instance of a probabilistic classifier. The output  $p(c|d)$  of a probabilistic classifier is the probability that the pattern  $d$  belongs to class  $c$  (posterior probability). The Naive Bayes classifier assumes that text data comes

from a set of parametric models (each single model is associated to a class). Training data are used to estimate the unknown model parameters. During the operative phase, the classifier computes (for each model) the probability  $p(d|c)$  expressing the probability that the document is generated using the model. The Bayes theorem allows the inversion of the generative model and the computation of the posterior probabilities (probability that the model generated the pattern). The final classification is performed selecting the model yielding the maximum posterior probability. In spite of its simplicity, the Naive Bayes classifier is almost as accurate as state-of-the-art learning algorithms for text categorisation tasks [14]. The Naive Bayes classifier is the most used classifier in many different web applications such as focused crawling, recommending systems, etc. For all these reasons, we have selected this classifier to measure the accuracy improvement provided by taking into account visual information.

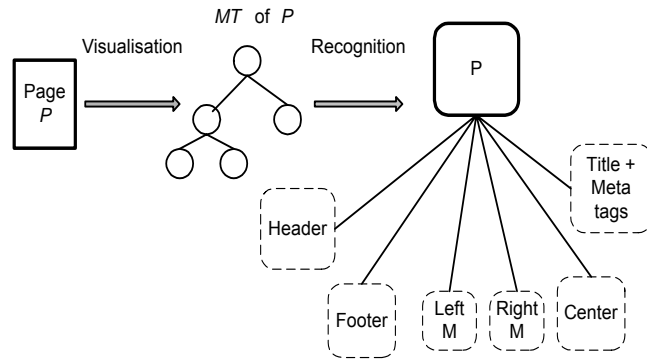
## 5.2 Classification results

The dataset was split into a training and a test set of equal size. First, a Naive Bayes classifier was trained on all the words in the documents. This classifier is usually constructed when not considering visual information and it provides a baseline to validate the effectiveness of the proposed data representation. In order to classify a page taking into account its visual appearance, each page from the training (test) set was processed, extracting the bag-of-words representation of its six basic constituents: header, footer, left menu, right menu, centre and title and meta-tags (see Figure 6). Then, we created six Naive Bayes classifiers where the  $i$ -th classifier was trained using the bag-of-words representations of the  $i$ -th constituents of the documents (i.e. the third classifier has been trained on the words belonging to the left menu of the documents). When classifying a document, the  $i$ -th classifier assigns a score to the document equal to  $p_i(c|d)$ . Mixing the scores of each single classifier we obtain the final decision. The mixture is performed assigning to the  $i$ -th classifier a weight  $w_i$  taking into account the expected relevance of the information stored into a specific part of the page:

$$p(c|d) = \sum_i w_i * p_i(c|d)$$

In particular, after some tuning we have assigned the following weights to each classifier: header 0.1, footer 0.01, left menu 0.05, right menu 0.04, centre 0.5, title and meta-tags 0.3 (weights are normalised and sum to 1). Table 2 shows the classification results of the proposed method. Taking into account the visual appearance of the page that is provided by *MT*, we achieved an improvement of more than 10% in the classification accuracy.

**Figure 6** Page representation used as an input for six Naïve Bayes classifiers



**Table 2** Comparison of the classification accuracy of a Naïve Bayes classifier when taking into account a bag-of-words representation of the page versus a mixture of Naive Bayes classifiers taking into account the visual appearance of documents. Information contained in the visual appearance increases classification accuracy by more than 10%

	<i>Correct (no visual)</i>	<i>Correct (visual)</i>	<i>Total</i>
Arts	105	176	324
Business	123	224	316
Computers	129	182	319
Games	141	212	339
Health	271	284	380
Home	284	269	348
Kids & Teens	81	171	343
News	198	218	336
Recreation	161	161	338
Reference	232	180	320
Science	121	163	335
Shopping	126	203	304
Society	194	180	337
Sports	145	222	341
Total	2311	2845	4680
	(49%)	(61%)	(100%)

## 6 Discussion

In recent years the internet has rapidly become a very valuable source of information about the competitive environment of companies and has been reported by a Futures Group survey in 1997 to be one of the top five sources for CI professional [15].

Competitive Intelligence (CI) is nothing else but the use of public sources to develop information about the competition, competitors and the market environment [16]. On corporate web sites one can find a lot of useful information concerning product overviews, business strategies, financial data, annual reports, company history, biographies of executives etc. This data is very important when inferring the competition's strategy for the future and thus can have an influence on our own. The internet (web) is essentially a free of charge source of information. That fact additionally attracts CI professionals. On the other hand data collection process is the most time-consuming task in CI projects (more than 30% of the total time spent) [17]. It would be much better if the time was spent in data analysis instead of data acquisition. Hence, we need intelligent tools for data acquisition and classification in logical categories.

Search engines are commonly used to find and download the information. Nevertheless, the exponential growth of the web makes it extremely difficult to index and locate interesting sites. General-purpose search engines tend to collect as much data as possible and to cover the entire web. Therefore, a lot of time passes between consecutive visits to the same web page by a search-engine's spider. This negative effect becomes more important as the size of the web grows. Since the content of the page dynamically changes, we end up with incorrect indexes (actually, when we search against the search engine's database, we look into the past state of the internet like when we look into the sky full of stars!). To overcome this currency problem we need specialised focused engines that are capable of finding new information in a short time by visiting only limited areas of interest. Such a system should be easily configurable for different domains of information and, therefore, needs some sort of intelligence to recognise the topic of the page. Also it should be intelligent enough to optimise visits to the nodes in the web graph. Presentational layout analysis helps in topic recognition (classification) as one can see in Section 5. We also hope it can improve crawling (visiting) strategies as was explained in Section 1.

## 7 Conclusions

This paper describes a possible representation for a web page in which objects are placed into a well-defined tree hierarchy according to where they belong in an HTML structure of a page. We named this representation *M-Tree*. Further, each object (node from *M-Tree*) carries information about its position in a browser window. This visual information enables us to define the heuristics for recognition of common areas such as header, footer, left and right menus, and centre of a page. The crucial difficulty was to develop a sufficiently good rendering algorithm i.e. to imitate the behaviour of popular user agents such as internet Explorer.

We concluded from the analysed pages that the HTML source was often far away from the proposed standard and it posed additional problems in the rendering process. After applying some techniques for error recovery in the construction of the parsing tree and introducing some rendering simplifications (we do not deal with frames, layers and style sheets) we defined recognition heuristics based only on visual information. We could have included other types of information into the recognition process, but we wanted to observe a percentage of successfully recognised areas based only on page layout structure and common design patterns. The overall success in recognising targeted areas yields 73%. From Table 1 one can see that menus are either recognised or not. On

the other hand, recognition of header and footer is more complex and heuristics other than just visual have to be considered. In further research, we plan to improve the rendering process and recognition heuristics. Also, we plan to recognise logical groups of objects that are not necessarily in the same area, like titles of figures, titles and subtitles in text, all commercial add-ins etc.

Initial results have shown that spatial information is important when classifying web documents. The classification accuracy of a Naive Bayes classifier was increased by more than 10%, when taking into account the visual information. In particular, we constructed a mixture of classifiers, each one trained to recognise words appearing in a specific portion of the page. In the future, we plan to use Neural Networks to find the optimal weights of our mixture of classifiers. We hope our system will also improve focused crawling strategies [4] by estimating the importance of the link based on its position and neighbourhood. We believe that our visual page representation can find its application in many other areas related to search engines, information retrieval and data mining from the web.

Also, initial experiments have shown that presentational layout analysis could improve the technologies needed for the construction of a domain-specific information acquisition system. Since this focused system would cover only a small portion of the web, it could amortise the currency problem associated with general-purpose search engines. On the other hand, such a system could be significantly less demanding in hardware and network resources. This would allow companies to maintain their own information repositories and to support the decision-making process in a cost-effective manner and independently of big players like Google or Yahoo.

## References and Notes

- 1 Quinlan, J.R. (1986) 'Induction of decision trees', *Machine Learning*, pp.81–106.
- 2 Salton, G. and McGill, M.J. (1983) *An Introduction to Modern Information Retrieval*, McGraw-Hill.
- 3 Chakrabarti, S., Van den Berg, M. and Dom, B. (1999) 'Focused crawling: a new approach to topic-specific web resource discovery', *Proceedings of the 8th Int. World Wide Web Conference*, Toronto, Canada.
- 4 Diligenti, M., Coetzee, F., Lawrence, S., Giles, C. and Gori, M. (2000) 'Focused crawling using context graphs', *Proceedings of the 26th Int. Conf. On Very Large Databases*, Cairo, Egypt.
- 5 Rennie, J. and McCallum, A. (1999) 'Using reinforcement learning to spider the web efficiently', *Proceedings of the Int. Conf. On Machine Learning*, Bled, Slovenia.
- 6 Bernard, L.M. (2001) 'Criteria for optimal web design (designing for usability)', Wichita State University, November, <http://psychology.wichita.edu/optimalweb/position.htm>
- 7 Embley, D.W., Jiang, Y.S. and Ng, Y.K. (1999) 'Record-boundary discovery in web documents', *Proceedings of SIGMOD*, Philadelphia, USA.
- 8 Lim, S.J. and Ng, Y.K. (1998) 'Extracting structures of HTML documents using a high-level stack machine', *Proceedings of the 12th International Conference on Information Networking ICIN*, Tokyo, Japan.
- 9 World Wide Web Consortium (W3C) (1999) 'HTML 4.01 specification', December, <http://www.w3c.org/TR/html401/>

- 10 James, F. (2001) 'Representing structured information in audio interfaces: a framework for selecting audio marking techniques to represent document structures', Stanford University, December, <http://www-pcd.stanford.edu/frankie/thesis/>
- 11 Siena Tree is written in Java 1.3 and can be used to visualize objects of interest from a web page. One can enter any sequence of HTML tags to obtain the picture (visualization) of their positions. To obtain a demo version contact [milos@grf.bg.ac.yu](mailto:milos@grf.bg.ac.yu)
- 12 The dataset can be downloaded from <http://nautilus.dii.unisi.it/download/webDataset.tar.gz>
- 13 Mitchell, T. (1997) *Machine Learning*, McGraw Hill.
- 14 Sebastiani, F. 'Machine learning in automated text categorization', *ACM Computing Surveys*, Vol. 34, No. 1, pp.1–47.
- 15 Futures Group (1998) 'Ostriches and Eagles 1997', The Future Group Articles.
- 16 McGonagle, J.J. and Vella, C.M. (1990) 'Outsmarting the competition', *Sourcebooks*, Naperville, USA.
- 17 Prescott, J.E. and Smith, D.C. (1991) 'SCIP: who we are, what we do', *Competitive Intelligence Review*, Vol. 2, No 1, pp. 3–5.