Automatic annotation of online articles based on visual feature classification

Radek Burget* and Ivana Burgetová

Faculty of Information Technology,
Brno University of Technology,
Bozetechova 2, 612 66 Brno, Czech Republic
E-mail: burgetr@fit.vutbr.cz
E-mail: burgetova@fit.vutbr.cz
*Corresponding author

Abstract: When applying the traditional data mining methods to World Wide Web documents, the typical problem is that a normal web page contains a variety of information of different kinds in addition to its main content. This additional information such as navigation, advertisement or copyright notices negatively influences the results of the data mining methods as for example the content classification. In this paper, we present a method of interesting area detection in a web page. This method is inspired by an assumed human reader approach to this task. First, basic visual blocks are detected in the page and subsequently, the purpose of these blocks is guessed based on their visual appearance. We describe a page segmentation method used for the visual block detection, we propose a way of the block classification based on the visual features and finally, we provide an experimental evaluation of the method on real-world data.

Keywords: automatic annotation; online articles; page segmentation; document preprocessing; visual features; visual analysis; data mining; classification.

Reference to this paper should be made as follows: Burget, R. and Burgetová, I. (2011) 'Automatic annotation of online articles based on visual feature classification', *Int. J. Intelligent Information and Database Systems*, Vol. 5, No. 4, pp.338–360.

Biographical notes: Radek Burget received his PhD in Information Technology in 2004 from the Brno University of Technology. He is an Assistant Professor at the Faculty of Information Technology, Brno University of Technology. His research interests include data mining methods, semi-structured data modelling, knowledge engineering and the semantic web.

Ivana Burgetová received her PhD in Computer Science and Engineering in 2009 from the Brno University of Technology. Currently, she is an Assistant Professor at the Faculty of Information Technology, Brno University of Technology. Her research focuses on bioinformatics, data mining and theoretical computer science.

Copyright © 2011 Inderscience Enterprises Ltd.

1 Introduction

In the last years, almost all the information published in the classical printed media is also available electronically via the World Wide Web. Newspapers, magazines and press agencies maintain their online versions and a large number of electronic magazines, weblogs and other types of online media make the web a vast source of electronically published articles.

It is the specific feature of the web that the articles themselves are usually buried in other content that forms the web page such as the web navigation, related articles, discussion or advertisement. As shown in Yi et al. (2003), this feature of web pages significantly complicates further application of data mining and information retrieval methods on the published articles. When applying the text classification methods on the whole web documents, the additional information in the page can be viewed as a noise that reduces the precision of the classification. Therefore, it is desirable to remove the additional content before further computer processing of the article. This requires the application of a detection algorithm that recognises the main article in the web page and eliminates the remaining information.

Similar situation occurs when automatically processing the article contents. The articles usually consist of several content blocks with a different meaning that include headings, author, publication date, lead paragraph and text paragraphs. The identification of these parts is also important for further article processing. For example, different weights may be assigned to different content blocks when a search index is being built.

Currently used web publication technology provides very limited options for an explicit annotation of the purpose of the individual content blocks and often, even these limited options are not used by the publisher. Therefore, the automatic identification of the article in the web page and the recognition of the purpose of its individual parts require a thorough analysis of the document. One of the possible ways is analysing the visual information that is provided for human readers.

The human readers usually use the visual features of the content for finding the main article in the page and for identifying the purpose of its individual parts. The designers of the web pages help the users by maintaining some style rules commonly accepted for this kind of contents. An article usually forms a visually separated block in the page with a different visual style than the surrounding information. Its main heading can be easily identified in the whole page thanks to its visual features, mainly the font size and style. The internal structure of the article is expressed visually as well. The leading paragraph, publication date, eventual subheadings and other blocks are clearly visually separated from the main text of the article.

Although the visual presentation of the document content is apparently important for human readers, most of the existing approaches to the automatic content block recognition are based on the analysis of the document code usually represented as a DOM tree. On the other hand, very few authors have examined the role of the actual visual features of the content. In this paper, we propose an approach to automatic annotation of important web page elements based on the above observations regarding the visual content presentation. We propose a model that describes the human perception of the page as exactly as possible. In order to use all the visual information that is available to the human reader, we work with a model of the rendered page instead of a more common approach based on modelling the document code. We propose detecting the basic blocks

in the page by an adopted page segmentation algorithm and subsequently, we propose a way of classification of the detected blocks based on their mutual positions and other visual features. As the result, we can assign a class to each detected block that helps to decide if the block forms a part of an article and eventually its meaning in the article. This information is potentially useful for further automatic processing of the documents.

We present the results of our experimental application of the proposed approach on real web documents. According to our experiments, the results can be used for detecting the important areas in web documents and the method can be used as an alternative to the existing DOM-based approaches or eventually, both approaches may be combined.

2 Related work

From a general point of view, our paper belongs to a broader area of information extraction from web documents. The research in this area has been running from the late '90s and during this time, many different approaches have been developed. Most of the methods are based on specialised procedures called *wrappers* that identify a particular information in a document based on recognising HTML code patterns that typically surround the desired information in a defined set of documents (Kushmerick, 1997). Since a manual construction of the wrappers is a time-consuming work, many methods have been proposed for automatic wrapper construction based on a set of annotated sample documents. The most important approaches include grammatical or automata inference (Freitag, 1997; Kosala et al., 2002) and relational machine learning (Cohen et al., 2002). The main problem of the wrapper is their sensitivity to changes in the documents (often called *brittleness*) which may cause the wrapper to stop working properly at any time (Kushmerick, 2000). Therefore, some approaches introduce more general models of the processed documents. This allows separating the extraction algorithm from the details of the document code. For example, Burget (2004) proposes a conceptual modelling approach. Some of the existing tools provide a human-assisted visual wrapper definition (Baumgartner et al., 2001; Liu et al., 2001) where the resulting wrappers are described using an abstract language in order to obtain greater flexibility.

The area of web page cleaning and information block detection in web pages can be viewed as a specific application of the general information extraction methods. Most of the existing approaches are based on the analysis of the HTML code of the document which is usually represented by a DOM (e.g., Chakrabarti et al., 2008; Gupta et al., 2003; Kovacevic et al., 2002; Lin and Ho, 2002; Mukherjee et al., 2003; Yi et al., 2003). The advantage of the DOM approach is generally its simplicity and scalability. On the other hand, the DOM itself gives an approximation of the resulting page only. For example, it does not contain any information about the positions of the individual content blocks and their visual style. In order to obtain the complete picture, the actual visual features of the content must be computed according to the attached style sheets, used fonts and other information which actually results in document rendering.

The approaches based on the rendered document processing usually work with a visual page representation obtained by page segmentation. The work published by Yu et al. (2002) is based on a visual page segmentation using the VIPS algorithm Cai et al. (2003). The closest to our approach is the work of Song et al. (2004) that is also based on the visual block classification. It assigns one of the four importance levels to each block based on its visual features. In contrast to this approach, we do not evaluate an overall

importance of the blocks. Instead, we assign more specific classes to the visual blocks in order to distinguish their purpose. Moreover, we have chosen a different set of visual features used for classification that we found more suitable as discussed in Section 5.

Our approach is based on a modification of our previously published page segmentation algorithm Burget (2007). Similarly to Song et al. (2004), we work with visual blocks detected in the rendered documents. However, our approach models the page division with finer granularity and it looks for areas consistent in their visual style. As the next step, we use the classification approach based on the visual features only for recognising the particular areas in the page. In contrast to the algorithms based on the HTML code analysis, there are no additional requirements on the underlying HTML code such as the usage of particular HTML elements. Moreover, the method is independent on the document language since only the visual features are used.

This paper is an extended version of a previously published paper Burget and Rudolfová (2009). In contrast to the original paper, we have introduced a new set of visual features used for the classification that allowed increasing the achieved precision significantly. We have greatly extended the experimental testing part by using larger sets of data and more classification methods. And lastly, we provide a more detailed description of the proposed approach and its evaluation.

3 Overview of the approach

Our approach is inspired by the way the human readers read the web pages. When looking for the article in a web page, the readers accomplish the following tasks:

1 identification of the basic visual blocks in the web page

2 classification of these blocks based on various features in order to determine their meaning

3 choosing the appropriate block that is worth reading.

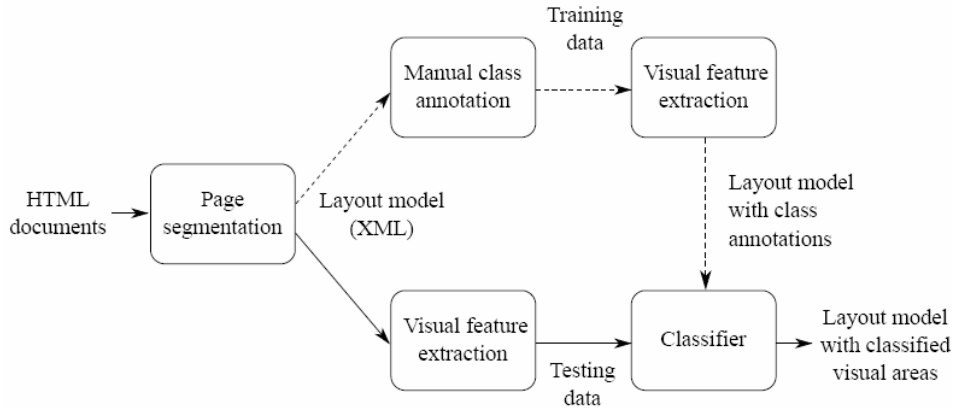
In this paper, we focus on the first two tasks. The detection of the basic visual blocks in the page is accomplished by employing a page segmentation algorithm. For accomplishing the second task, various visual attributes of the detected blocks are extracted and a classification algorithm is used for assigning classes to the individual blocks.

The whole visual area classification process is illustrated in Figure 1. It consists of two phases – a training phase and the classification phase itself.

In both phases, the first step is page segmentation. In this step, the page is rendered and basic visual blocks are detected. The details are described in Section 4. As the result, we obtain a model of the page layout stored in an XML file. The whole page is represented as a tree of visual blocks that describe the page layout from the root block (the whole page) to the smallest detected blocks. For each block, the model contains its position and other visual properties.

For the training phase, we segment a sample set of web pages and we manually annotate the visual blocks that form part of a contained article such as headings, text paragraphs, etc. Then, we extract the visual properties of all the visual blocks in the sample set of web pages and use them as the training examples for the classifier together with the manual annotations.

Figure 1 A schema of the visual area classification process



Note: Dashed lines show the training phase, solid lines show the classification phase.

In the classification phase, we process new HTML documents that have not been manually annotated. Each document is segmented using the same algorithm and the same visual features of the detected blocks are extracted directly from the obtained layout model. Obtained data forms an input of the previously trained classifier that assigns classes to the individual blocks.

The details of both the training set preparation and the used classification algorithms are provided in Section 5.

4 Page segmentation approach

The purpose of the page segmentation is to detect the visual blocks that can be assigned a meaning in context of the article (for example a heading or a paragraph). Therefore, we are looking for separated groups of text lines in the page where the whole group has a consistent visual style.

The segmentation method we have used is based on our previously published method (Burget, 2007) with significant modifications. We have redefined the purpose of the box clustering phases so that the algorithm consists of a *line detection phase* and a subsequent *block detection phase* that includes the visual style comparison. As the result, we obtain a larger number of smaller areas with consistent style from the segmentation process. This result is more suitable for the following step of area classification described in Section 5.

The input of the page segmentation algorithm is an HTML document and the related Cascading style sheets required for rendering the document. The output is a tree of detected areas that models the visual properties of the areas and their eventual nesting. The whole segmentation process consists of the following steps:

1 page rendering – obtaining the information about the positions of the basic document elements on the page and about their visual style.

2 detecting basic visual areas – detecting the elements that form visual areas in the page and creating a tree of areas based on their nesting

3 text line detection – joining the areas on the same line

4 block detection – detecting the larger areas with the same visual style of the blocks

In the following sections, we briefly describe the individual phases.

4.1 Page rendering

For rendering the web page, we have used an experimental *CSSBox* layout engine (http://cssbox.sourceforge.net/) that has been designed mainly for this purpose. The task of the rendering engine is to load the HTML code of a document and all the corresponding CSS style sheets and subsequently, to determine the positions and visual features of all the elements on the resulting page. The way of determining these features is given mainly by the CSS specification (Bos et al., 1998).

The rendering is performed on a virtual canvas with the initial size of 1,000 × 800 pixels which may be adjusted according to the resulting style (computed widths and heights) of the rendered elements. The result of the rendering is (according to the CSSBox terminology) a set of *boxes*. By a box, we understand a rectangular area in the resulting page with a given position and size on the canvas containing an arbitrary part of the document content. Generally, there is a box created for each DOM element (so called *element boxes*), for each connected text string in the document (*text boxes*) and for each inserted object such as an image (so called *replaced boxes*). For each box, the following visual features are computed by the rendering engine:

- size and position on the page – the *x* and *y* coordinates and the *width* and *height*; we will call this the *box bounds*
- background colour (some boxes may have transparent background)
- font properties (font size, weight and style)
- border properties – in CSS, there may be a border (a frame) of an arbitrary width defined around a box
- if the box is formed by a text, we obtain the text string.

For further analysis, each box may be represented as a tuple:

$$b = (type, x, y, width, height, bgcolor, fontsize, weight, style, border, text) \tag{1}$$

where *type* is either the element box, text box or replaced box as defined above and the other members correspond to the above mentioned visual features. All the positions and sizes are measured in pixels. The background colour (*bgcolor*) is represented as a RGB value or a special *transparent* value, if there is no background colour defined for the given box. The *weight* and *style* properties have the value of 0 for normal font and 1 for bold or italic font respectively.

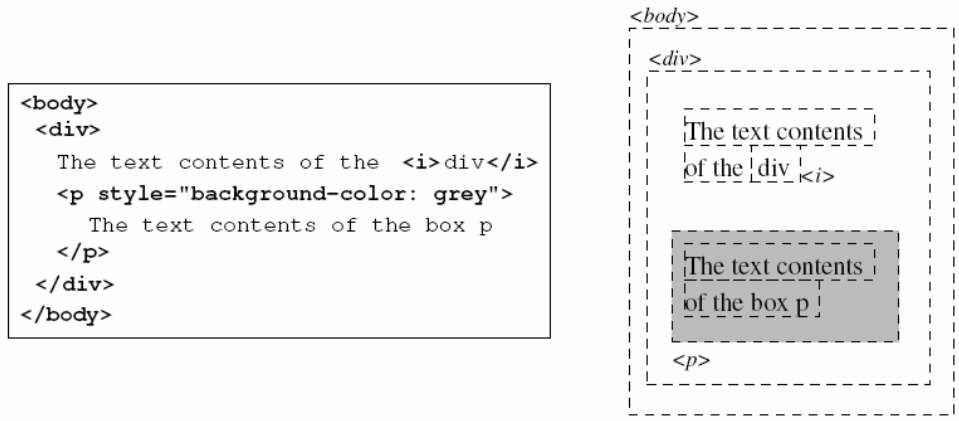
The whole rendered document may be then represented as a set of boxes

$$B = \{b_1, b_2, b_3, ..., b_n\} \tag{2}$$

where *n* is the total number of boxes in the document. Subsequently, this set of boxes is analysed in the next phase of the page segmentation algorithm in order to discover the visual blocks in the page.

Figure 2 shows an example of a simple HTML document and the rendered page. The bounds of the individual boxes are marked with dashed lines. As we may see, more text boxes may be generated for a continuous text because of the automatic line wrapping that is applied according to the CSS specification.

Figure 2 An example of an HTML document and the boxes obtained by document rendering



4.2 Detecting basic visual areas

In HTML documents, one of the following means may be used for creating visually separated blocks in the page:

- using a box that is visually separated from the remaining content, e.g., by a different background colour or a frame
- creating groups of boxes that are separated from the remaining content by other visual means that include line separators and rivers of white space.

In this phase of the page segmentation, we detect the boxes from the set B that form standalone visual blocks, i.e., the first one of the above mentioned cases. We say that a box is *visually separated* if at least one of the following conditions holds:

- the box is created by the document root element and thus, it represents the whole rendered page
- the box is a text box or a replaced box as defined in Section 4.1. These boxes represent the actual document contents
- the background of the box is not transparent or there is a visible border defined around the box, i.e., the box is apparent on the resulting page.

Each box that meets the above definition is perceived by the user as a basic standalone unit of the page and we will call it a *basic visual area*. The bounds of this area correspond to the bounds of the appropriate box. Let $a(b_i)$ be a basic visual area that corresponds to the box $b_i \in B$. In this step, we create a set of basic visual areas:

$$A = \{a(b_i); b_i \in B \wedge b_i \text{ is visually separated}\} \tag{3}$$

In many cases, the areas are overlapping in the page. A typical example is a colour box in the page that contains several lines of text. Each text line is represented by a text box $b_i \in B$ and it is placed inside of the bounds of an element box $b_j \in B$; $i \neq j$ with a colour background. We say, that the text boxes b_i are fully enclosed in the background box b_j. In our example in the Figure 2, this is the case of the paragraph <*p*> and its content boxes. In HTML documents, this is a very frequent situation that is simply created by two or more nested HTML elements with the appropriate visual properties.

Very rarely, two boxes may be detected in the document that overlap only partially; than means, the areas overlap but we cannot say that one of the areas is fully enclosed in the other one. In this case, we consider the box drawing order defined in the CSS specification and we say that the box b_i is enclosed in b_j when b_i is drawn in front of b_j according to the CSS specification.

We model the box overlapping by creating a tree of visual areas:

$$T_A = (A, E) \tag{4}$$

where A is the set of basic visual areas (3) and E is the set of tree edges, where $(a(b_j), a(b_i)) \in E$ if and only if b_i is enclosed in b_j.

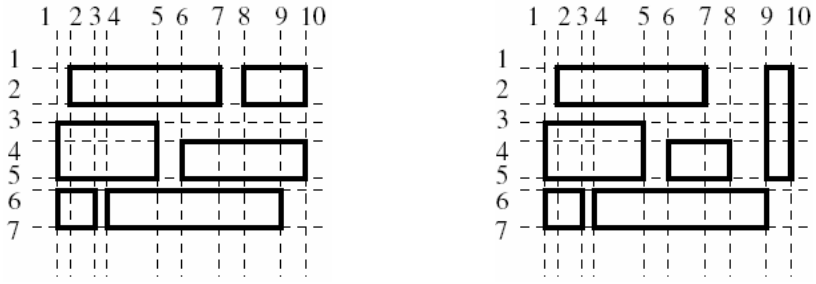
The resulting tree represents the visual area nesting as perceived by the user. Its root area represents the whole page (it corresponds to the document root element) and the leaf areas correspond to the individual pieces of text or other content. The remaining areas correspond to other boxes that are visually separated for example by a colour border or background.

The tree of basic areas obtained from the sample document from the Figure 2 is shown in the Figure 4(a). The <*div*> and <*i*> boxes are not visually separated and thus, they do not create a visual area. The <*p*> box is separated by a colour background and the remaining boxes are the text boxes and the document root.

4.2.1 Area grid

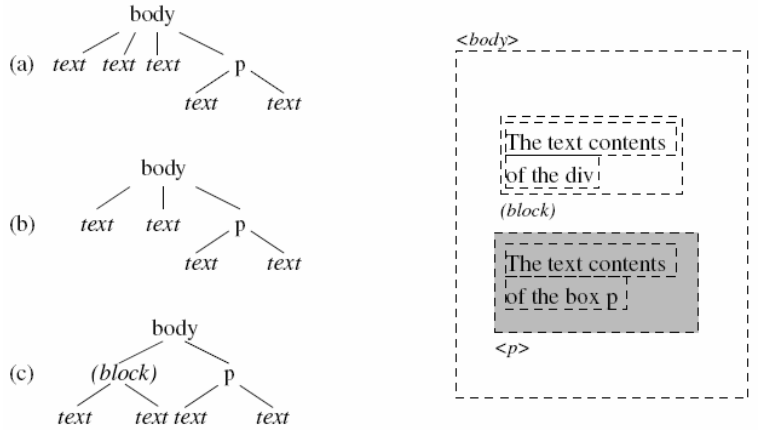
The above defined tree of visual areas represents the area nesting. For the further steps of the page segmentation, it is also necessary to represent the mutual positions of the areas within their parent area. For this purpose, we define an area grid g_i for each area $a(b_i) \in A$ that describes the mutual positions of all the child areas of $a(b_i)$ in T_A. Examples of such a grid are shown in the Figure 3.

Figure 3 Examples of two area grids used for line detection (a) with three lines detected, (b) with a box preventing the first two lines from being detected



(a) (b)

Figure 4 The basic visual area tree, (a) the resulting tree after the line detection phase (b) the block detection phase (c) the resulting bounds of the detected visual areas



All the child areas of $a(b_i)$ are placed in a grid with variable column widths and row heights where each area occupies an arbitrary number of rows and columns. The position of each area is determined by its starting and ending column and row in the grid. Using this representation, we can quickly determine the mutual positions of an arbitrary pair of child areas. Similar grids are created for all the child areas recursively. These grids are primarily used for locating the directly neighbouring areas in the page and for the line and block detection as described in next chapters.

4.3 Text line detection

The purpose of this phase is to join the areas that form a single text line. In our approach, we consider a text line to be the smallest visual area that is not broken to even smaller pieces. In order to detect the lines, we look for the areas that share the same rows in the appropriate area grid and the bounds of the line do not overlap with any other detected line or existing area. When found, we check if the areas are not visually separated from each other. In this case, we require that the areas have the same background colour and there is no separating border between the areas.

If the areas are not visually separated, we join them into a single area representing the whole line. The areas $a(b_i)$, $a(b_j) \in A$ that share the same parent area in T_A will be replaced by a new area $a(b_i, b_j) \in A$. The bounds of the resulting area then cover all the joined areas and the area corresponds to a set of boxes instead of a single box. The set of child areas of the new area in T_A is the union of the sets of child areas of $a(b_i)$ and $a(b_j)$.

For example, in the Figure 3, we detect three lines in the grid (a): the first line spans for the grid rows (1, 2), the second one is (3, 5) and the third one is (6, 7). In contrast, in the grid (b), the first two lines will not be detected because they would overlap with the box in the column 9. This corresponds to the fact that the areas form a more complicated structure that does not correspond to simple lines of text.

In our example in the Figure 4, the text ‘of the div’ is created by two boxes that correspond to the two elements in the original document. After applying the line detection algorithm, these two boxes are joined to a single one. The resulting tree of visual areas is shown in the Figure 4(b).

4.4 Block detection

The last step of the page segmentation is the detection of visually consistent blocks. In this phase, we create new areas in T_A that group together adjacent areas with the same style. The purpose of this step is to detect the paragraphs of a consistent style (most frequently the headings consisting of multiple lines or the simple paragraphs of text) that can be later classified as standalone areas. Two areas are considered to have a consistent style if they share the same average font size and style of their text contents.

Moreover, we consider the visual separators that may appear in the page and that can be created by other visually separated areas or by rivers of white space. For the detection of these separators, we use the block division algorithm published in Cai et al. (2003).

For each area, we have a set of separators and a set of child areas with the given style. When detecting the covering areas, we find the largest rectangular areas that cover the child areas with the same style. At the same time, we require that the new areas do not cover any of the separators. When such a covering area is found, it is added to the area tree at the appropriate point so that the tree still represents the area nesting.

In our example, the first two lines of text have a consistent style and they are not visually separated by any separator. We create a new area that covers these text lines. The resulting tree of areas is shown in the Figure 4(c) where the newly created area is marked as *block*.

4.5 Segmentation result

As the result of the above described segmentation process, we obtain a tree of areas, where the root area corresponds to the whole page; the leaf areas correspond to the individual text lines and the remaining areas corresponds to some visually distinguished blocks in the page. This model is built based on both the visual properties of the page elements and various kinds of separators. Therefore, the obtained tree represents the visual page organisation as it is perceived by a user.

For the purpose of its further analysis, we serialise the obtained tree of areas to an XML file. This file contains the description of all the detected visual areas including their positions in the page and in the layout grid described in Section 4.2.1. For the leaf visual areas, the information about contained text boxes and their visual properties is included as well. Thus, each XML file contains all information necessary both for displaying the original page contents and for determining the visual properties of each detected area.

5 Page element classification

All the non-leaf nodes of the visual area tree obtained from the page segmentation correspond to some visual areas detected in the page. For each of these visual areas, we determine the values of various visual properties that are used for the area classification. The task of the classifier is to assign a class to each area based on these properties.

The classification process consists of two phases. In the learning phase, we train the classifier using a training set of segmented documents where the classes have been manually assigned to the individual areas. Then, in the classification phase, new

documents are segmented and the discovered areas are assigned classes using the previously trained classifier.

5.1 Obtaining the visual features of the areas

For the classification of large sets of documents from different sources, it is important that the chosen visual features be comparable for the whole set of previously unknown documents. Therefore, we have avoided using absolute values such as absolute font size and we prefer relative values of all features.

For obtaining the values of all the features, we have implemented a *feature extraction tool*. This software tool reads a set of XML files containing a segmentation result for a set of documents as described in Section 4.5 and it produces a file in the ARFF format that can be used as an input for the WEKA classifier described below. This file contains a list of all the visual areas in the document set where each area is represented by the values of the individual features.

The following visual features are computed for all the visual areas of all the processed documents.

5.1.1 Font features

The font features are computed as the average values for the whole visual area evaluated weighted by the number of characters with a particular value of that feature. The following values are computed:

- *fontsize* – average font size in percent where the average font size of the whole document is considered to be 100%. Thus, the value tells whether the font size for a particular area is below or above average.
- *weight* – an average font weight from the range 0..1. The regular font characters are assigned the weight of 0; bold characters have the weight of 1.
- *style* – an average font style from the range 0..1 (normal or italic style) computed analogically to the average weight.

5.1.2 Spatial features

Spatial features describe the position of the given area in the page and the relations to other areas.

- *aabove*, *abelow*, *aleft*, *aright* – the number of areas that are placed above, below, on the left and on the right of the area within its parent area.
- *relx*, *rely* – the relative position of the area within the whole page. 0 means the left edge or top of the page respectively, 1 means the right edge or bottom.

5.1.3 Text features

The text content of an area is a text string obtained as a concatenation of all the text boxes covered with the area. The following values are computed for each area:

- *nlines* – number of text lines in the area.

- *tlength* – total length of the contained text in characters.
- *pdigits*, *plower*, *pupper*, *pspaces*, *ppunct* – percentages of digits, lowercase letters, uppercase letters, whitespaces and punctuation characters in the text content of the area.

5.1.4 Colour features

Colour features include the colour properties of the area background and the average colour properties of the text. In order to obtain some comparable values, we do not consider the colours themselves but we compute the following values:

- *tlum* – an average text luminosity computed according to the WCAG recommendation (Caldwell et al., 2008). The appropriate formula in this recommendation considers the real properties of the human vision and therefore, it seems to be suitable for this purpose.
- *bglum* – the background colour luminosity computed using the same formula. If the given area has a transparent background, we consider the background colour of its parent area.
- *contrast* – the average colour contrast computed from the text and background luminosities according to the WCAG recommendation.
- *cperc* – the percentage of text of the same colour in the document. This value tells how much this text colour is unique or prevalent within the given document.

5.2 Training data preparation

For preparing the training set of documents, we have implemented a graphical *annotation tool* that allows displaying a segmented page stored in an XML file and assigning a particular class to the displayed visual areas. The assigned class is then stored back to the XML file. The extraction tool can then include the assigned class in the input data for the classifier together with the computed feature values. The classifier input obtained this way is then used as the training dataset. This process corresponds to the training phase illustrated in Figure 1.

Moreover, the graphical annotation tool can be used for displaying the visual feature values of the individual detected areas and after the classification phase is finished, it is able to display the results obtained by the automatic classification and eventually, to compare them graphically with the manually assigned classes.

The overview of the classes used for annotation is in Table 1. These classes correspond to the individual parts of articles that are commonly used in internet sources. A special class *none* is automatically assigned to all remaining areas detected in the page that do not form part of the article contents.

It is a special feature of the dataset prepared this way that the number of areas assigned to the *none* class significantly exceeds the number of areas assigned to the other classes. This corresponds to the fact that in a normal web page, most of the detected areas do not correspond to any standalone part of the articles. Many of them correspond to the single text lines that form only a part of some paragraph, some areas correspond to other type of additional content.

Table 1 The classes assigned to the individual visual areas

`h1`	Main article heading
`h2`	Second-level heading in the article
`subtitle`	The article subtitle
`perex`	The leading paragraph of the article
`paragraph`	An ordinary paragraph
`date`	Publication date
`author`	Author name
`authordate`	Author and the date in a single area
`none`	Remaining areas that do not belong to the article

5.3 Classification algorithms

For the classification itself, we have used the University of Waikato WEKA data mining tool (http://www.cs.waikato.ac.nz/˜ml/index.html) that offers a variety of implemented classification methods based on various principles. WEKA can read both the training and testing datasets produced by our software tools and it also allows evaluating the results of classification, mainly the precision and recall rates for the individual classes.

We have tested the following classification methods for our data:

- the J48 tree classifier which is an implementation of the Quinlan's (1993) C4.5 algorithm
- Bayesian network classifier
- multilayer perceptron – a neural network with backpropagation
- support vector machines.

The advantage of the J48 algorithm is that it produces a decision tree that is interesting for our further research. Therefore, we have used this algorithm in our previous works and we include it for comparison in this work. Bayesian network presents a classical classification method. We have chosen it for its reliable results in our previous experiments. The neural network classifier and the SVM classifier have been chosen for comparison with the results published by Song et al. (2004).

We have experimentally compared these algorithms on datasets we have created from real web data. All the selected methods produced acceptable results. The detailed data about our experiments are provided in section 6.

6 Experimental evaluation

For the experimental evaluation, we have implemented several tools according to the Figure 1 introduced in the overview. The segmentation tool implements the page segmentation algorithm described in Section 4. For each segmented document, the resulting tree of visual areas is serialised to XML and stored to a separate file. Secondly, the graphical annotation tool allows to display the XML files obtained from segmentation and to manually assign classes to the individual areas. And finally, the feature extraction

tool processes a set of segmented pages (either annotated or not annotated), it computes the values of all the visual features for all the visual areas contained in these files and stores the obtained values into an ARFF file that may be used as a training or testing dataset for the WEKA classifier. All our tools have been implemented in the Java environment.

We have tested our classification approach on a set of web documents obtained from the websites of various newspapers worldwide. We segmented all the downloaded documents using our segmentation tool and we manually annotated a subset of documents from each website. Subsequently, we have computed the visual features of all the visual areas detected in all the documents and tested the classification algorithms on various subsets of these data. As the next step, we have tested selected algorithms in situations that correspond to the assumed real use of the method.

Table 2 The websites used as sources of the testing documents and the corresponding RSS feed URLs

Website	*RSS feed*
Aktualne.cz	http://aktualne.centrum.cz/export/rss-hp.phtml
Idnes.cz	http://servis.idnes.cz/rss.asp?c=zpravodaj
Novinky.cz	http://novinky.cz/rss2/
Lupa.cz	http://rss.lupa.cz/2/clanky/
Root.cz	http://www.root.cz/rss/clanky/
Lidovky.cz	http://www.lidovky.cz/export/rss.asp?r=ln domov
El Mundo	http://rss.elmundo.es/rss/descarga.htm?data2=4
El Pais	http://www.elpais.com/rss/feed.html?feedId=1022
La Vanguardia	http://feeds.feedburner.com/lavanguardia/alminuto
LA Times	http://feeds.latimes.com/latimes/news?format=xml
USA Today	http://rssfeeds.usatoday.com/usatoday-NewsTopStories
Zeit Online	http://newsfeed.zeit.de/index
Focus Online	http://rss.focus.de/fol/XML/rss folnews.xml
Stern.de	http://www.stern.de/feed/standard/all/
Le Figaro	http://rss.lefigaro.fr/lefigaro/laune?format=xml
Le Parisien	http://rss.leparisien.fr/leparisien/rss/actualites-a-la-une.xml
20minutes.fr	http://www.20minutes.fr/rss/flux/une.xml
Corriere Della Sera	http://www.corriere.it/rss/homepage.xml
La Repubblica	http://rss.feedsportal.com/c/32275/f/438637/index.rss
Il Sole 24 ore	http://feeds.ilsole24ore.com/c/32276/f/438662/index.rss

6.1 Source data

We have manually created a list of source websites that is provided in Table 2. Each of these websites provides an RSS feed that contains the URLs of the latest published articles. We have automatically segmented all the articles in the RSS feeds. From each

website, from five to 15 documents have been obtained according to the different sizes of the RSS feeds. In total, we have obtained 559 documents and during the segmentation, 198,129 visual areas have been detected in these documents.

Using the annotation tool, we have manually annotated four documents from each website. In total, we have manually annotated 784 areas in these documents. From the annotated documents, we have created various training and testing datasets for testing the classification method in different situations. The remaining segmented documents have been used for visual evaluation of the classification results using our graphical tool.

We have used the obtained datasets for two basic experiments:

1 We have compared the precision and recall of the selected classification algorithms listed in 5.3 in two different situations regarding the combination of training a testing sources. The aim of these experiments was to check the fitness of the individual algorithms for the given purpose. These experiments are further discussed in the Section 6.2. Based on the obtained values, we have chosen the SVM and BayesNet classifiers for further experiments.

2 With the selected classification methods, we have run thorougher experiments in order to verify the classification results for the expected use cases of our approach. We have tested more different combinations of the training and testing sets in order to minimise the influence of possible specific features of the individual web sources. These experiments are discussed in the Section 6.3.

6.2 Classification algorithm evaluation

According to our previous experiments, the resulting precision and recall of the individual classification algorithms greatly depends on the actual combination of the datasets used for training and testing. This is caused mainly by the difference in the way of the visual article presentation among various web sources where some sources are more similar to each other than the other ones. Therefore, the first step of our experimental evaluation was to find the classification algorithms that give satisfactory results for both the favourable combinations, where the way of visual presentation is similar in the training and testing data sources and the unfavourable combinations, where the visual presentation differs significantly. Based on the obtained results, we choose the algorithms that should be the most suitable for the application on real use-cases that are described in the Section 6.3.

For comparing the classification algorithms, we have created two datasets:

- The first dataset is used for testing the algorithms in an ‘optimistic’ situation when the training and testing documents come from the same sources. Both the training and testing sets contain two annotated documents from each source in Table 2, however, different documents from these sources have been used for creating the training and testing sets.
- The second dataset simulates a ‘pessimistic’ variant when the training set of pages is created from different sources than the testing set. We have randomly split the set of sources in two parts of the same size and we have used all the annotated documents from these sources to create the training and testing sets respectively.

These datasets should represent the favourable and unfavourable conditions for the classification. Now, our task is to choose the algorithms that give satisfactory results for both datasets.

Table 3 Obtained precision, recall and F-measure values of the classification algorithms for the individual classes

Class	*J48*			*BayesNet*		
	P	*R*	*F*	*P*	*R*	*F*
none	0.986	0.992	0.989	0.996	0.936	0.965
h1	0.727	0.889	0.800	0.400	0.889	0.552
h2	1.000	0.737	0.848	0.673	0.921	0.778
subtitle	0.500	0.143	0.222	0.081	0.429	0.136
perex	0.455	0.714	0.556	0.524	0.786	0.629
paragraph	0.743	0.663	0.701	0.386	0.961	0.550
author	0.200	0.091	0.125	0.273	0.273	0.273
date	1.000	0.222	0.364	0.292	0.778	0.424
authordate	1.000	0.300	0.462	0.600	0.600	0.600
	Perceptron			*SVM*		
	P	*R*	*F*	*P*	*R*	*F*
none	0.988	0.982	0.985	0.991	0.986	0.988
h1	0.889	0.889	0.889	0.842	0.889	0.865
h2	0.906	0.763	0.829	0.886	0.816	0.849
subtitle	0.500	0.286	0.364	0.500	0.143	0.222
perex	0.417	0.714	0.526	0.524	0.786	0.629
paragraph	0.612	0.795	0.691	0.674	0.787	0.726
author	0.000	0.000	0.000	0.417	0.455	0.435
date	0.000	0.000	0.000	0.500	0.556	0.526
authordate	0.000	0.000	0.000	0.700	0.700	0.700

Note: The training and testing dataset are coming from the same sources (optimistic variant).

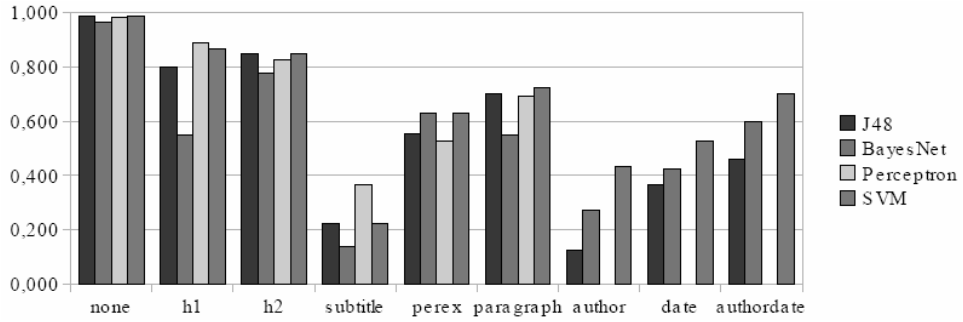
Table 3 shows the classification results for the first dataset. We include the values of precision, recall and the F-measure value computed as

$$F = \frac{2 \cdot P \cdot R}{P + R}$$

The graphical comparison of the F-measure values for the individual classification methods and classes is in Figure 5. We can see that all the methods are able to identify most visual areas that belong to the *h1*, *h2*, *paragraph* and *perex* classes. Such visual areas are contained in most of the annotated documents and they usually have some

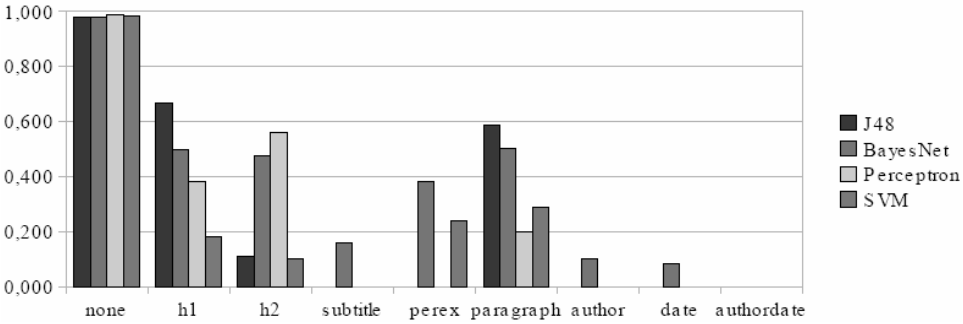
specific visual features that make them easy to recognise. On the other hand, some methods fail for the remaining classes such as *author* or *date*. These visual areas are often not contained in the documents and the way of their presentation is very variable. The *subtitle* class is used in very few web sources and therefore, it is rarely identified correctly. SVM classifier gives stable results for all the classes, the J48 and BayesNet classifier work for most classes too.

Figure 5 Comparison of the F-measures of the algorithms for individual classes (optimistic variant)



The results for the pessimistic variant are quite different. In this case, the visual properties of the individual area classes are much more different than in the optimistic case. The obtained values are shown in Table 4 and the graphical comparison in Figure 6. In this situation, only the BayesNet classifier was able to identify at least some visual areas from all the classes; other methods fail mainly for the classes with more variable presentation.

Figure 6 Comparison of the F-measures of the algorithms for individual classes (pessimistic variant)



The results show that none of the classification methods can be considered to be the most suitable one for the classification of visual areas. However, SVM gives reliable results in situations where the content is presented in a consistent manner. On the other hand, BayesNet classifier is usable even when the content presentation style varies significantly.

For these reasons, we have selected both the SVM and the BayesNet classifiers for running more tests that simulate the expected use cases of the proposed method.

Table 4 Obtained precision, recall and F-measure values of the classification algorithms for the individual classes

Class	*J48*			*BayesNet*		
	P	*R*	*F*	*P*	*R*	*F*
none	0.991	0.967	0.979	0.996	0.960	0.978
h1	0.714	0.625	0.667	0.341	0.938	0.500
h2	0.079	0.176	0.109	0.429	0.529	0.474
subtitle	0.000	0.000	0.000	0.095	0.500	0.160
perex	0.000	0.000	0.000	0.250	0.813	0.382
paragraph	0.487	0.736	0.586	0.375	0.757	0.501
author	0.000	0.000	0.000	0.063	0.300	0.103
date	0.000	0.000	0.000	0.043	1.000	0.083
authordate	0.000	0.000	0.000	0.000	0.000	0.000
	Perceptron			*SVM*		
	P	*R*	*F*	*P*	*R*	*F*
none	0.977	0.998	0.987	0.978	0.994	0.986
h1	0.308	0.500	0.381	0.333	0.125	0.182
h2	0.875	0.412	0.560	0.333	0.059	0.100
subtitle	0.000	0.000	0.000	0.000	0.000	0.000
perex	0.000	0.000	0.000	0.333	0.188	0.240
paragraph	0.810	0.115	0.201	0.537	0.196	0.287
author	0.000	0.000	0.000	0.000	0.000	0.000
date	0.000	0.000	0.000	0.000	0.000	0.000
authordate	0.000	0.000	0.000	0.000	0.000	0.000

Note: The training and testing dataset are coming from different sources (pessimistic variant).

6.3 Results for expected use cases

The aim of the following experiment was to test the method behaviour for two typical use cases of our proposed method of automatic article annotation:

1 The training documents come from the same source as the later annotated documents. In this case, the method is used for identifying the articles or parts of articles coming from a single, previously known source.

2 There is a fixed set of training documents obtained from larger number of sources and the documents being automatically annotated come from other, previously unknown source or sources. This can be the case of sources where no training examples are available in advance.

We consider the first case to be more probable in practical application since the article sources are usually known in advance. However, the second case may occur too; for example when the data sources may be dynamically added by users or when using the method for processing ad hoc documents found on the web.

The main difference from the experiments previously discussed in Section 6.2 is that we have tested more combinations of sources for creating the training and testing sets in order to eliminate possible specific features of the individual web sources.

In the first case, only a single source is considered. Therefore, we had to increase the number of annotated documents. We have annotated 20 documents coming from the same source and we have used 6 documents for creating the training set and the rest of the documents for testing. We have repeated this for four different sources. Table 5 shows the average results for the individual classes.

Table 5 Achieved results for the training documents coming from the same source as the later annotated documents

Class	*BayesNet*			*SVM*		
	P	*R*	*F*	*P*	*R*	*F*
none	0.999	0.917	0.957	1.000	0.999	0.999
h1	1.000	1.000	1.000	1.000	1.000	1.000
h2	0.727	1.000	0.842	1.000	1.000	1.000
subtitle	0.500	1.000	0.667	1.000	1.000	1.000
perex	0.286	1.000	0.444	0.800	1.000	0.889
paragraph	0.381	0.971	0.547	0.986	0.986	0.986
author	0.273	1.000	0.429	0.750	1.000	0.857
date	0.200	0.750	0.316	1.000	0.500	0.667
authordate	0.571	1.000	0.727	1.000	1.000	1.000

Since all the documents from a single source usually maintain a consistent style of presentation, we can see that both methods give very reliable results. In this case the SVM classifier gives better results that the BayesNet classifier.

For simulating the second use case, we have used the annotated documents from one source as a testing set and the documents from the remaining sources as the training set. Again, we have repeated the test for four different combinations of sources. The results are shown in Table 6.

Table 6 Achieved results for documents coming from other, previously unknown source

Class	*BayesNet*			*SVM*		
	P	*R*	*F*	*P*	*R*	*F*
none	0.998	0.879	0.934	0.971	0.998	0.984
h1	1.000	0.750	0.857	0.667	0.500	0.571
h2	0.833	0556	0.667	0.800	0.444	0.571
subtitle	0.254	0.928	0.399	0.000	0.000	0.000
perex	0.400	1.000	0.571	0.136	0.300	0.187
paragraph	0.247	0.899	0.388	0.972	0.507	0.667
author	0.250	0.667	0.364	0.000	0.000	0.000
date	0.500	0.333	0.400	0.000	0.000	0.000
authordate	0.125	0.167	0.143	0.000	0.000	0.000

In this case, the BayesNet method gives more reliable results. Generally, the results are acceptable for more common classes such as headings and paragraphs. The values for

less frequent and differently presented classes such as author and date are quite low for the reasons mentioned above. The results in both cases overcome our previous results published in Burget and Rudolfová (2009) where we have used a lower number of visual features for the visual area classification.

6.4 Comparison with other approaches

In order to compare our results with the results published in Song et al. (2004), we have followed the same scenario as described in the paper. With all the annotated data, we have conducted a five-fold cross validation using the SVM classification method.

Table 7 shows the results of the cross validation for our dataset. In Song et al. (2004), the classes are defined in a more general way according to the relevance of the given content block to the article: Actual parts of the article are marked with *Level 4*, other areas of the page relevant to the document topic or useful for the user are marked with *Level 3* and *Level 2* and the remaining parts (noisy information) are marked with *Level 1*. For the classification, the levels 2 and 3 are regarded as a single class. Since in our dataset, we have only annotated the parts of the article itself, all our classes different from *none* correspond to *Level 4* and the *none* class corresponds basically to the levels 1 to 3. We have mapped the classes as described. The comparisons of the obtained results with the results of Song et al. (2004) are in the Table 8.

Table 7 Results of the five-fold cross validation for our method by classes

Class	*P*	*R*	*F*
none	0.995	0.997	0.996
h1	0.795	0.861	0.827
h2	0.767	0.836	0.800
subtitle	1.000	0.417	0.588
perex	0.826	0.731	0.776
paragraph	0.912	0.907	0.910
author	0.706	0.414	0.522
date	0.778	0.667	0.718
authordate	0.750	0.882	0.811

We can see that the results for the *Level 4* class are comparable or slightly better for our method. The results for the remaining classes are more difficult to compare because our *none* class does not directly correspond to a single class in the reference paper. However, the results demonstrate that the visual features of the document content can be used for automatic content annotation with a sufficient precision.

Table 8 Comparison of the five-fold cross validation results

Class	*Our method*			*Song et al. (2004)*		
	P	*R*	*F*	*P*	*R*	*F*
Level 1	0.995	0.997	0.996	0.763	0.776	0.769
Level 2				0.796	0.804	0.800
Level 4	0.917	0.873	0.895	0.839	0.770	0.803

7 Applications and possible improvements

The proposed method is suitable for pre-processing the documents before their further automatic indexing, classification or other processing by a computer. It can be used for cleaning the document from the noisy information or for supplying the further processing tools with an additional information about the individual parts of the document structure.

The results show that our approach is able to automatically annotate visual areas in a page with quite a high precision at least when applied to a single source of documents. In that case, it is able to correctly annotate most of the repeating parts of the articles.

When used on new, previously unknown sources, the method can be used for detecting the most common parts of the article such as headings. For certain applications such as the page preprocessing for data mining mentioned in the introduction, the obtained precision and recall of the classification seems to be satisfactory. This precision allows to determine most of the visual areas that belong to the article and to detect the approximate bounds of the main article on the page.

In order to determine the bounds of articles in previously unknown documents more precisely, the method could be extended by using a kind of heuristics that would represent some commonly used habits in the article publication on the web. For example, in Burget (2009), we have proposed and tested some heuristics concerning the layout of the text placed below a header.

Since only visual attributes of the contents are used, the presented approach is independent on the document language. Regarding the properties of the text contents of the document, our approach is only based on some statistical properties of the text. As the next step, the visual features could be combined with the traditional text classification methods by analysing for example the term frequencies in the individual visual areas. However, this would introduce the dependency on a particular language.

8 Conclusions

In this paper, we have introduced a method of interesting area detection in the web pages. We have proposed a method of page segmentation tailored especially for this purpose in order to detect the basic visual blocks in the page that have a consistent visual style. Further, we have proposed the way of the detected block classification based on a set of their visual features. Finally, we have tested the proposed method on real-world data.

According to the results of the experiments, the method is suitable for an approximate detection of a published article in the web page. In case of using only one, previously known source of documents, the obtained precision allows a reliable identification of article elements in the page.

In comparison to the methods based on document code (DOM) analysis, our approach allows extracting more different visual features of the document contents, especially regarding the spatial and colour features. As shown by the experimental results, this allows achieving an interesting precision while preserving independence on the document language. However, both the visual features and the DOM model can be arbitrarily combined. The combination of the visual features with the information obtained from the DOM model seems to be a promising way for improving the quality of the page segmentation and the classification itself.

Acknowledgements

This work was partially supported by the BUT FIT grant FIT-S-10-2 and the research plan MSM0021630528.

References

Baumgartner, R., Flesca, S. and Gottlob, G. (2001) 'Visual web information extraction with lixto', *VLDB '01: Proceedings of the 27th International Conference on very Large Data Bases*, Morgan Kaufmann Publishers Inc., pp.119–128.

Bos, B., Lie, H.W., Lilley, C. and Jacobs, I. (1998) 'Cascading style sheets, level 2', *CSS2 Specification: The World Wide Web Consortium*.

Burget, R. and Rudolfová, I. (2009) 'Web page element classification based on visual features', *IEEE Computer Society, 1st Asian Conference on Intelligent Information and Database Systems ACIIDS 2009*, pp.67–72.

Burget, R. (2004) 'Hierarchies in HTML documents: linking text to concepts', *IEEE Computer Society, 15th International Workshop on Database and Expert Systems Applications*, pp.186–190.

Burget, R. (2007) 'Layout based information extraction from HTML documents', *IEEE Computer Society, ICDAR 2007*, pp.624–629.

Burget, R. (2009) 'Automatic web document restructuring based on visual information analysis', *Proceedings of the 6th Atlantic Web Intelligence Conference – AWIC'2009*, Vol. 10, Springer Verlag.

Cai, D., Yu, S., Wen, J-R. and Ma, W-Y. (2003) 'VIPS: a vision-based page segmentation algorithm', *Microsoft Research*.

Caldwell, B., Cooper, M., Reid, L.G. and Vanderheiden, G. (2008) 'Web content accessibility guidelines 2.0', *The World Wide Web Consortium*.

Chakrabarti, D., Kumar, R. and Punera, K. (2008) 'A graph-theoretic approach to webpage segmentation', *17th International World Wide Web Conference*.

Cohen, W.W., Hurst, M. and Jensen, L.S., (2002) 'A flexible learning system for wrapping tables and lists in html documents', *ACM, WWW '02: Proceedings of the 11th international conference on World Wide Web*, pp.232–241.

Freitag, D. (1997) 'Using grammatical inference to improve precision in information extraction', *ICML-97Workshop on Automata Induction, Grammatical Inference, and Language Acquisition*.

Gupta, S., Kaiser, G., Neistadt, D. and Grimm, P. (2003) 'DOM-based content extraction of HTML documents', *WWW2003 proceedings of the 12 Web Conference*, pp.207–214.

Kosala, R., Bussche, J.V.D., Bruynooghe, M. and Blockeel, H. (2002) 'Information extraction in structured documents using tree automata induction', *PKDD '02: Proceedings of the 6th European Conference on Principles of Data Mining and Knowledge Discovery*, pp.299–310, Springer-Verlag.

Kovacevic, M., Diligenti, M., Gori, M. and Milutinovic, V. (2002) 'Recognition of common areas in a web page using visual information: a possible application in a page classification', *IEEE Computer Society, ICDM '02*, p.250.

Kushmerick, N. (1997) Wrapper induction for information extraction', PhD thesis.

Kushmerick, N. (2000) *Wrapper Verification: World Wide Web*, Vol. 3, No. 2, pp.79–94.

Lin, S-H. and Ho, J-M. (2002) 'Discovering informative content blocks from web documents', *ACM, KDD '02: Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp.588–593.

Liu, L., Pu, C. and Han, W. (2001) 'An XML-enabled data extraction toolkit for web sources', *Inf. Syst*., Vol. 26, No. 8, pp.563–583.

Mukherjee, S., Yang, G., Tan, W. and Ramakrishnan, I., (2003) 'Automatic discovery of semantic structures in HTML documents', *IEEE Computer Society, International Conference on Document Analysis and Recognition*.

Quinlan, J.R. (1993) *C4.5: Programs for Machine Learning*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Song, R., Liu, H., Wen, J-R. and Ma, W-Y. (2004) 'Learning block importance models for web pages', *ACM, WWW '04: Proceedings of the 13th international conference on World Wide Web*, pp.203–211.

Yi, L., Liu, B. and Li, X. (2003) 'Eliminating noisy information in web pages for data mining', *ACM, Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*.

Yu, S., Cai, D., Wen, J-R. and Ma, W-Y. (2002) 'Improving pseudo-relevance feedback in web information retrieval using web page segmentation', *Microsoft Research*.