# レイアウトツリーによるウェブページ ビジュアルブロック表現方法の提案

曾駿[†1]　Brendan Flanagan[†1]　廣川佐千男[††2]

ウェブページに対する情報抽出には類似する要素を抜き出すためにパターン認識が行われている。しかし、従来の手法には HTML ファイルのソースコードを分析することにより要素を抽出する方法が多い。これらの手法は言語依存であるので、克服できない欠点がある。本論文ではレイアウトツリーという視覚的に類似する特徴を持つページブロックを認識する方法と提案する。本稿でウェブページで各要素が表示される長方形領域をビジュアルブロックと言う。これらのブロックに含まれる要素が表示される相対位置をレイアウトツリーとして表す。２つのブロックのレイアウトツリーの類似度によりこの２つのブロックの相似度を求める。

# Proposal of Layout Tree of Web Page as Description of Visual Blocks

JUN ZENG[†1]　BRENDAN FLANAGAN[†1]　SACHIO HIROKAWA[††2]

When extracting information from a web page, IE systems usually need to perform pattern recognition to identify the elements that have similar patterns. However, most of them are mainly based on analyzing HMTL source code, DOM tree, tag tree or Xpath of web pages. These methods are language-dependent, or more precisely, HTML-dependent. They have some insuperable limitations. In order to overcome these limitations, we propose a notion of layout-tree and a pattern recognition method to identify visual blocks with similar visual pattern using layout tree. In this paper, we call a visible rectangular region in a web page a visual block or block for short. We consider if the elements of two blocks are displayed in a similar layout, we define that the two blocks are visually similar. We first transform the layout into a layout tree. By calculating the similarity of the layout trees of two blocks, we can determine whether the two blocks are visually similar or not.

## 1. Introduction

The Web, as the largest database, often contains information that may be interesting for researchers and the general public. However, the quantity of information available today is more than at any point in history, but with this wealth of information comes even greater challenges. Unlike structured database, Web pages are semi-structured data. Due to the lack of structure of web information sources, access to this huge collection of information has been limited to extracting and searching automatically. The process that extracts information from semi-structured data (such as web pages) and translates the information into structured data is called Information Extraction (IE) [1].

Programs that perform IE task are referred to as extractors or wrappers. Wrappers are used to identify data of interest and map them to some suitable format such as, XML or relational tables. In order to identify the elements with similar pattern, a wrapper usually performs pattern recognition which rely on a set of extraction rules. These rules are mainly based on analyzing the HMTL source code, Document Object Model (DOM) tree [2, 3], tag tree [4, 5] or Xpath [6, 7] of web pages. These methods have some insuperable limitations. They depend on web page programming languages. With the reversion of these languages, some new tags will be introduced and some tags may be deprecated. For example, by comparison with HTML4, HTML5 adds many new syntactic features, such as <video>, <audio> and <canvas> elements. Also, some elements, such as <a>, <cite> and <menu> have been changed. Once the version of languages changes, these methods are not able to adapt to the new version of the language. Thus the life circle of these methods is very short.

In order to overcome these limitations, vision-based methods have been proposed [8, 9]. These methods rely on visual cues from browser renderings. Most of the vision-based methods focus on the location, size or font features of elements. However, these approaches can only be applied to certain web page templates. For example [9] clusters the data records through analyzing the similarity of position, image size and font size of the elements and consider that the main contents or data records are always in the middle of web page. Even though such assumptions are important for the success of the algorithm, it is hard to see how the proposed approaches can be used for pages with other semantic structures.

Besides the HTML structure and visual cues, there is another important feature that is often ignored. That is relative position of elements. Some may consider relative position as a visual property. Strictly speaking, relative position is different from

†1 九州大学大学院 システム情報科学府
Graduate School of ISEE, Kyushu University
†2 九州大学 情報基盤研究開発センター
Research Institute for Information Technology, Kyushu University

other visual properties. The visual properties such as position, size and font size only refers to just one single element, but relative position refers to at least two elements. In other words, the visual properties like position, size and font size are absolute and a single tuple, but layout is relative and therefore a double tuple. The relative positions of the elements can form the layout of these elements. In this paper, we translate the layout feature into a tree structure called layout tree and propose a pattern recognition method to identify visual block (see Section 3 for definition) with similar visual pattern using layout tree.

The rest of the paper is organized as follows: Related works are reviewed in Section 2. Visual block and layout feature are introduced in Section 3. Our solution for identify visually similar block is described in Section 4. Finally, conclusion and future work are given in Section 5.

## 2. Related Work

In the past few years, many approaches to information extraction have been proposed. According to the pattern recognition rules, we roughly divide these approaches into two groups: HTML-based approaches and vision-based approaches.

### 2.1 HTML-based Approaches

HTML source code is often transformed into three forms: DOM tree, tag tree, xpath and tag path. These approaches identify the similar patterns by calculating the similarity or distance of DOM tree, tag tree, xpath or tag path of elements.

(1) **DOM Tree**

Analyzing the DOM tree is a basic and effective way to identify the structure feature of HTML documents. D. Yuan et al. [2] consider the nodes labeled "div" as topic content node, which may contain the important information. They prune back the noise nodes which are not topic node and extract the information from the pruned DOM tree. C. Castillo et al. [3] defined the length of the path between two nodes of a DOM tree as DOM distance. This method is based on DOM distance and can extract information from single webpages or collections of interconnected webpages.

(2) **Tag Tree**

The tag tree can be regarded as a simplified DOM tree. It focuses on the tag name of HTML elements and igores the other attributes and properties. X. Ji et al. [4] parsed web pages into tag trees, and then generated templates using a cost-based tree similarity measurement. The exclusive content in each page is then extracted by using the templates to parse the page. Finally, the records in pages and the schema of the records can be extracted from the exclusive content by finding repeating patterns and using some heuristic rules. X. Xie et al. [5] transfered a distinct group of tag paths appearing repeatedly in the DOM tree of the Web document to a sequence of integers, from which a suffix tree is built by using this sequence. Then they captured the useful data region patterns which can be used to extract data records.

(3) **Tag Tree**

Similar to the tag tree, the Xpath or tag path makes use of the tag name of HTML elements to analyze the structure of HTML documents. G. Miao et al. [6] focuses on how a distinct tag path appears repeatedly in the DOM tree of a Web document. Instead of comparing a pair of individual segments, a pair of tag path occurrence patterns were compared to estimate how likely these two tag paths represent the same list of objects. T. Grigalis et al. [7] mainly segment a web page using xpath. This method clusters visually and structurally similar repeating web page elements to identify the underlying data records.

All of these approaches are language-dependent, or more precisely, HTML-dependent. Once the version of languages changes, these methods are not able to adapt to the new version of the language. Moreover, an HTML document is just one part of a web page. Web pages also need the support of some script languages, such as: Javascript and Cascading Style Sheets (CSS). Although, these script languages have little semantic function, they play an irreplaceable role and contain a lot of valuable information. In other words, an HTML file cannot be equated with a web page. Simply analyzing the web page programming language may lead to incorrect results.

### 2.2 Vision-based Approaches

Vision-based approaches rely on visual cues from browser renderings. Most of the vision-based methods focus on the location, size or font features of elements. These approaches can make good use of the visual information that is defined by Javascript or CSS.

J. Kang et al. [8] proposed an informative block extraction approach. This approach relies on visual clue for vision-based page block segmentation to analyze and partition a web page into a set of visual blocks, and then groups related blocks with similar content structures into block clusters by using a tree edit distance method. Then it recognizes the informative block cluster by using tree alignment and tree matching. W. Liu et al. [9] proposed a vision-based IE method that primarily utilizes the visual features on the deep Web pages to implement deep Web data extraction, including data record extraction and data item extraction.

However, these approaches can only be applied to certain web page templates and often need to make some assumptions. As these assumptions are integral to the success of the algorithm, it is hard to see how the proposed approaches could be used for

pages with other semantic structures.

## 3. Visual Similar Blocks

### 3.1 Definition of Visual Block

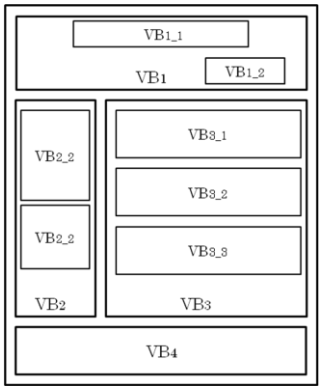



図 1　ビジュアルブロックの構造

Figure 1　The structure of visual blocks.

A web page is made up of finite blocks. We also call these blocks visual block or block for short. We consider a visual block as a visible rectangular region on a web page, as shown in Figure 1. The definition of a visual block is as follows:

**Definition III-1**: Visual block $VB = (E, R)$, where $E$ is an Element object that is defined by the HTML DOM based on W3C standard, and $R$ represents the visible rectangular region where $VB$ is displayed in web page.

According to W3C standard, the Element object of the DOM represents an element in the HTML document. The details of Element object can be found in official website of W3C [10]. The Element object not only contains the attributes of an HTML element, such as "tagName", "id", "value" etc., but also contains the properties defined by the DOM, such as "childNodes", "nextSibling", etc.

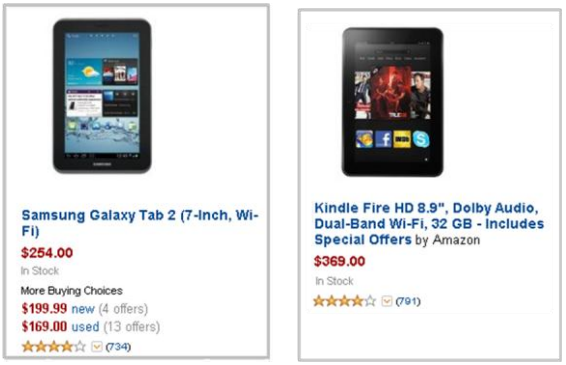
**Definition III-2:** For two given visual blocks $VB_1 = (E_1, R_1)$ and $VB_2 = (E_2, R_2)$, if $E_1$ is a descendant node of $E_2$, then $VB_2$ includes $VB_1$, denoted $VB_1 \subset VB_2$.

**Definition III-3:** If a visual block $VB = (E, R)$ does not include any other visual blocks, then $VB$ is a leaf visual block, denoted $VB$ : *leaf*.

### 3.2 Visually Similar Blocks

**Definition III-4:** For two given visual blocks $VB_1$ and $VB_2$, if the leaf visual blocks of $VB_1$ and $VB_2$ are displayed in a similar layout, we define that $VB_1$ and $VB_2$ are visually similar.

It should be noted that only the layout of leaf blocks are considered. This is because the leaf blocks contain contents such as text, images etc. The other intermediate visual blocks do not contain content, so they are ignored. Figure 2 shows two records of tablet computers. Although the contents of two records are not all the same, the main layout is similar. In both two records, a picture is on the top of records; the product names are under the pictures; the prices are under the product names; evaluations are on the bottom of records. The (a) record contains some additional contents, but the layout of "picture", "name", "price" and "evaluation" is the same in both (a) and (b). According to Definition III-4, (a) and (b) are visually similar blocks.




(a)　　(b)

図 2　（a）と（b）は視覚的に類似するブロックである

Figure 2　(a) and (b) are two visually similar blocks.

## 4. Layout Tree of Visual Block

### 4.1 The Layout of Visual Blocks

In this section, a description of layout is given and the creation of layout tree is introduced. For a visual block $B$, where $B$ is not a leaf block, the layout of $B$ is represented as a two-tuples Layout($B$) = ($LB$, $S$). $LB = \{b_i \,|b_i : leaf\ and\ b \subset B, i \in [1, n]\}$ is a finite sequence of leaf blocks that are included by $B$. All these blocks are not overlapping. The order of the leaf blocks are determined by depth-first traversal of the DOM tree. $S = \{s_1, s_2, \ldots, s_{n-1}\}$ is a finite sequence of separators, including horizontal separators and vertical separators. The direction of a separator is a simple and effective way to describe the relative position. If the separator is horizontal, it means the relative position of the two parts that are on the two sides of the separator is up-down. If the separator is vertical, it means the relative position is left-right. It should be noted that a separator never crosses any blocks.

Figure 3 shows an example of the layout of a visual block. In Figure 3, the solid line rectangles represent the leaf blocks and dotted lines represent the separators. All the intermediate blocks are ignored, because if they are considered the visual blocks may overlap each other, which will make it difficult to determine the separators. Therefore only the leaf blocks are considered to describe the layout of a visual block.

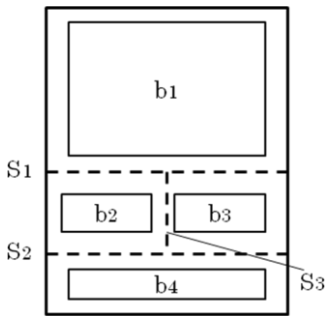



図 3　ビジュアルブロックのレイアウト

Figure 3　An example of the layout of a visual block.

### 4.2 The Layout Tree of Visual Block

The separators can be considered as nodes of a tree, and the two smaller parts can be considered as the left sub-tree and the right sub-tree. Generally, if the separator is horizontal, the upper part is left sub-tree and lower part is the right sub-tree. If the separator is vertical, the left part is left sub-tree and right part is as a tree. We call the tree a "layout tree". In this section, we introduce how to determine each separator and generate a layout tree.

We take the visual block in Figure 3 for example to explain the process of generating a layout tree as shown in Figure 4. Let us suppose that the ordered set of the leaf blocks $\{b_1, b_2, b_3, b_4\}$ have been figured out. First, $\{b_2, b_3, b_4\}$ are considered as a whole. There is a separator $S_1$ between $b_1$ and $\{b_2, b_3, b_4\}$. In Figure 4 (a) the first separator $S_1$ splits the block into two parts $P_1$ and $P_2$. Then the $S_1$ is considered as the root, the upper part $P_1$ is the left sub-tree and the lower part $P_2$ is the right sub-tree. After that, the two sub-trees are checked to see if contain a separator. In Figure 4 (b), $P_1$ contains only the leaf block $b_1$ and does not contain any separators. There is no need to separated $P_1$ anymore, so $P_1$ is replaced by $b_1$. The right sub-tree $P_2$ contains three leaf blocks $\{b_2, b_3, b_4\}$, so it needs to be separate further. Similarly, $\{b_3, b_4\}$ could be considered as a whole, however, there are not any separators between $b_2$ and $\{b_3, b_4\}$. Therefore $\{b_2, b_3\}$ is considered as a whole as there is a separator $S_2$ between $\{b_2, b_3\}$ and $b_4$. $S_2$ separates $P_2$ into two smaller parts. The upper part $P_{2\_1}$ is the left sub-tree and the lower part $P_{2\_2}$ is the right sub-tree. In Figure 4(c), $b_4$ replaces the $P_{2\_2}$ that is because $b_4$ is the only one leaf block that is contained in $P_{2\_2}$. $P_{2\_1}$ is separated by $S_3$ into $P_{2\_1\_1}$ and $P_{2\_1\_2}$. Finally, $P_{2\_1\_1}$ is replaced by $b_2$ and $P_{2\_1\_2}$ is replaced by $b_3$ as both $P_{2\_1\_1}$ and $P_{2\_1\_2}$ contain only leaf block. Figure 4(d) shows the final layout tree of the visual block.

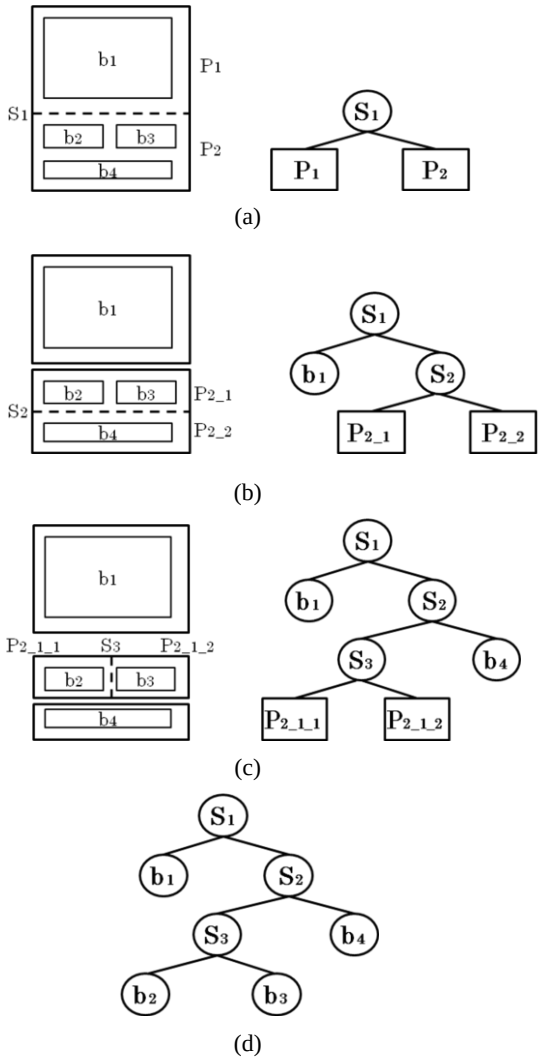



図 4　レイアウトツリーの生成

Figure 4　The process for layout tree generation.

### 4.3 Weight of Layout Tree

The contribution of different leaf blocks to the layout is different. For example in Figure 3, $b_1$ is more important than any other leaf blocks. If $b_1$ disappeared then the layout would change a lot. Conversely, if $b_2$ or $b_3$ disappeared the change of the layout is much less. Here, we call this contribution or importance "weight". For a leaf block $b_i$ the Weight($b_i$) is calculated as in formula (1) :

$$\mathrm{Weight}(b_i) = \frac{\mathrm{Area}(b_i)}{\mathrm{Area}(B)} \qquad (1)$$

Here $B$ is the visual block where $b_i$ is in, Area($b_i$) and Area($B$) represent the area of $b_i$ and $B$. In other words, in the same visual block, the leaf block with greater area has greater weight.

Similar to the leaf blocks, each separator has weight. It is not hard to notice that each separator can separate the current rectangular region and leaf blocks into two smaller parts. Therefore the weight Weight($S_i$) of a separator $S_i$ is calculated as formula (2):

$$\mathrm{Weight}(S_i) = \frac{Min\{\mathrm{Area}(P_1), \mathrm{Area}(P_2)\}}{\mathrm{Area}(B)} \qquad (2)$$

Here $B$ is the visual block containing $S_i$, $P_1$ and $P_2$ are the two smaller parts that are separated by $S_i$. Area() represents the area. Let us go back to the example of Figure 3, it is obvious that the order of the weight of the three separators is Weight($S_1$) > Weight($S_2$) > Weight($S_3$). Particularly, if Area($P_1$) + Area($P_2$) = Area($B$) and Area($P_1$) = Area($P_2$), the Weight($S_i$) will be the maximum value *1/2*.

### 4.4 Similarity of Layout Trees

According to Definition III-4, if two blocks has a similar layout feature, they are visually similar blocks. The similarity of layout trees can be regarded as the similarity of blocks. There are many algorithms to calculate the structural similarity between trees, in which the Tree Edit Distance (TED) is a simple and efficient algorithm [11]. We apply the TED algorithm to measure the similarity between layout trees. The edit distance, ED($T_1$, $T_2$), between two trees $T_1$ and $T_2$ is defined as the minimum cost to transform $T_1$ to $T_2$ by using insertion, deletion, and replacement operations on nodes. See paper [11] for the detail of TED algorithm.

Basing on the TED algorithm and the features of layout tree, we introduce the cost functions to calculate the cost of operations. Formula (3) and formula (4) show the cost functions of insertion and deletion operations:

$$\text{Insert}(n) = \text{Weight}(n) \quad (3)$$

$$\text{Delete}(n) = \text{Weight}(n) \quad (4)$$

Here $n$ is a node of a layout tree, and Weight($n$) is the weight of $n$. That is to say if insert $n$ into a tree or delete $n$ from a tree the cost will be the weight of $n$. The greater the weight is the greater the cost will be. Similarly, the cost function of a replacement operation is calculated as in formula (5)

$$\text{Re}(n_1, n_2) = \begin{cases} 0 & (n_1 \ sim \ n_2) \\ \text{Weight}(n_1) + \text{Weight}(n_2) & (n_1 \ diff \ n_2) \end{cases} \quad (5)$$

Here $n_1$ *sim* $n_2$ represents $n_1$ and $n_2$ are similar, and $n_1$ *diff* $n_2$ represents $n_1$ and $n_2$ are not similar. As introduced in previous section, there are two types of nodes in a layout tree: separator nodes and leaf block nodes. Moreover, there are two directions in separator nodes: horizontal and vertical. As for leaf block nodes, we roughly divide them into two types: image nodes and text nodes. The following rules are used to determine whether $n_1$ and $n_2$ are similar or not:

**Rule 1:** If node $n_1$ and node $n_2$ are different types (one is a separator node and the other is a leaf block node), then $n_1$ *diff* $n_2$.

**Rule 2:** If both node $n_1$ and $n_2$ are separator nodes, and the directions of $n_1$ and $n_2$ are different (one is horizontal and the other one is vertical) then $n_1$ *diff* $n_2$. Otherwise, $n_1$ *sim* $n_2$.

**Rule 3:** If both node $n_1$ and $n_2$ are leaf block nodes, and the types of $n_1$ and $n_2$ are different (one is image node and the other one is text node) then $n_1$ *diff* $n_2$.

**Rule 4:** If both node $n_1$ and $n_2$ are image nodes, then $n_1$ *diff* $n_2$.

**Rule 5:** If both node $n_1$ and $n_2$ are text nodes, and $n_1$ and $n_2$ have the same font and font size, then $n_1$ *sim* $n_2$. Otherwise $n_1$ *diff* $n_2$.

After the edit distance of two layout trees are determined, the similarity of them can be calculate. Let $T_1$ and $T_2$ be two layout trees. ED($T_1$, $T_2$) is the edit distance of $T_1$ and $T_2$. The similarity of $T_1$ and $T_2$ can be calculated as in formula (6):

$$\text{Sim}(T_1, T_2) = \frac{\text{ED}(T_1, T_2)}{Max\{\sum \text{Weight}(n_i), \sum \text{Weight}(m_i)\}} \quad (6)$$

Here $n_i$ is a node in $T_1$ and $m_i$ is a node in $T_2$. The denominator of formula (6) represents the greater weight of the layout tree $T_1$ and $T_2$. The similarity of $T_1$ and $T_2$ has the following features:

(1) $\text{Sim}(T_1, T_2) \in [0,1]$

(2) If Sim($T_1$, $T_2$) is closer to 0, then $T_1$ and $T_2$ are have a greater similarity; if Sim($T_1$, $T_2$) is closer to 1, then $T_1$ and $T_2$ are have greater different. We introduce a threshold $\alpha$. If Sim($T_1$, $T_2$) $\leq \alpha$, then $T_1$ and $T_2$ are similar, otherwise they are different.

## 5. Conclusion and Future Work

When extracting information from web pages, IE systems usually need to perform pattern discovery to identify the elements that have similar patterns. However, most of the pattern recognition methods are mainly based on analyzing HMTL source code, such as: DOM tree, tag tree or Xpath of web pages. These methods are language-dependent, or more precisely, HTML-dependent. They have some insuperable limitations. In order to overcome these limitations, we proposed a pattern recognition method to identify visual blocks with similar visual patterns using layout tree. In this paper, we call a visible rectangular region on a web page a visual block or block for short. We consider if the elements of two blocks are displayed in a similar layout, we define that the two blocks are visually similar. We used the separators to transform the layout of a block into a tree structure called layout tree. By calculating the

similarity of the layout trees of two blocks, we can determine whether the two blocks are visually similar or not.

The layout-tree-based pattern recognition method can be applied to many fields, such as: information extraction, pattern reorganization and data mining. In the future, we plan to develop an IE system to extract data records from web pages using the layout tree method.

## REFERENCES

1) C. Chang, M. Kayed, M. Girgis, and K. Shaalan, "A Survey of Web Information Extraction Systems", IEEE Transactions on Knowledge and Data Engineering, Vol. 18, Issue 10, pp. 1411-1428 (2006)
2) D. Yuan, Z. Mo, B. Xie, and Y. Xie, "The Technology of Extracting Content Information from Web Page Based on DOM Tree", Communications in Computer and Information Science Volume 144, pp. 271-278 (2011)
3) C. Castillo, H. Valero, J. Ramos, and J. Silva, "Information extraction from webpages based on DOM distances", Lecture Notes in Computer Science Volume 7182, pp. 181-193 (2012)
4) X. Ji, J. Zeng, S. Zhang, and C. Wu, "Tag tree template for Web information and schema extraction", Expert Systems with Applications Volume 37, Issue 12, pp. 8492–8498 (2010)
5) X. Xie, Y. Fang, Z. Zhang and L. Li, "Extracting data records from web using suffix tree", Proc. ACM SIGKDD Workshop on Mining Data Semantics (MDS '12), Article No. 12 (2012)
6) G. Miao, J. Tatemura, W. Hsiung, A. Sawires, and L. E. Moser, "Extracting data records from the web using tag path clustering", Proc. the 18th international conference on World wide web(WWW 09), pp.981-990 (2009)
7) T. Grigalis and A. Čenys, "Generating Xpath Expressions for Structured Web Data Record Segmentation", Communications in Computer and Information Science Volume 319, pp. 38-47 (2012)
8) J. Kang and J. Choi, "Recognising informative web page blocks using visual segmentation for efficient information extraction", Journal of Universal Computer Science, Vol. 14, No. 11, pp. 1893-1910 (2008)
9) W. Liu, X. Meng, and W. Meng, "Vide: A vision-based approach for deep web data extraction", IEEE Transactions on Knowledge and Data Engineering, Vol. 22, Issue 3, pp. 447-460 (2010)
10) http://www.w3.org/
11) K. Zhang and D. Shasha, "Simple Fast Algorithms for the Editing Distance between Trees and Related Problems", SIAM J. COMPUT, Vol. 18, No, 6, pp. 1245-1262 (1989)