Extraction of Informative Blocks from Deep Web Page Using Similar Layout Feature

Zeng, Jun
Graduate School of Information Science and Electrical Engineering, Kyushu University

Flanagan, Brendan
Graduate School of Information Science and Electrical Engineering, Kyushu University

Hirokawa, Sachio
Research Institute for Information Technology, Kyushu University

Extraction of Informative Blocks from Deep Web Page Using Similar Layout Feature

[1] Jun Zeng, [2] Brendan Flanagan, [3] Sachio Hirokawa
[1,2] *Graduate School of Information Science and Electrical Engineering, Kyushu University,*
[1]*zeng.j.000@s.kyushu-u.ac.jp*
[2]*bflanagan.kyudai@gmail.com*
[3] *Research Institute for Information Technology, Kyushu University,*
hirokawa@cc.kyushu-u.ac.jp

Abstract

Due to the explosive growth and popularity of the deep web, information extraction from deep web page has gained more and more attention. However, the HTML structure of web page has become more complicated, making it difficult to recognize target content by only analyzing the HTML source code. In this paper, we propose a method to extract the informative blocks from a deep web using the layout feature. We consider the visual rectangular region of an HTML element as a visual block in web page. We transform the elements' layout of a visual block into a layout tree. By calculating the similarity of layout trees, we cluster the visual blocks that have similar layout feature. Finally, the cluster which has the largest area is extracted as the informative block cluster. The experiment results show that this method is optimal when the threshold of layout tree similarity is 0.4.

Keywords: *layout tree, informative block, deep web page, information extraction.*

1. Introduction

The Deep Web (also called the hidden Web) differs from the surface Web which can be indexed by standard search engines [1]. The pages of deep web (we refer to these pages as deep web pages) are generated dynamically from a database. Due to the explosive growth and popularity of the deep web, information extraction for deep web page has gained more and more attention. Currently, most of the Information Extraction (IE) methods are mainly based on analyzing the HMTL source code. However, the HTML structure of a web document has become more complicated, making it more difficult to recognize the target content by analyzing the HTML source code only. Moreover HTML-based methods are language-dependent. Once the version of languages changes, these methods are not able to adapt to the new version of the language.

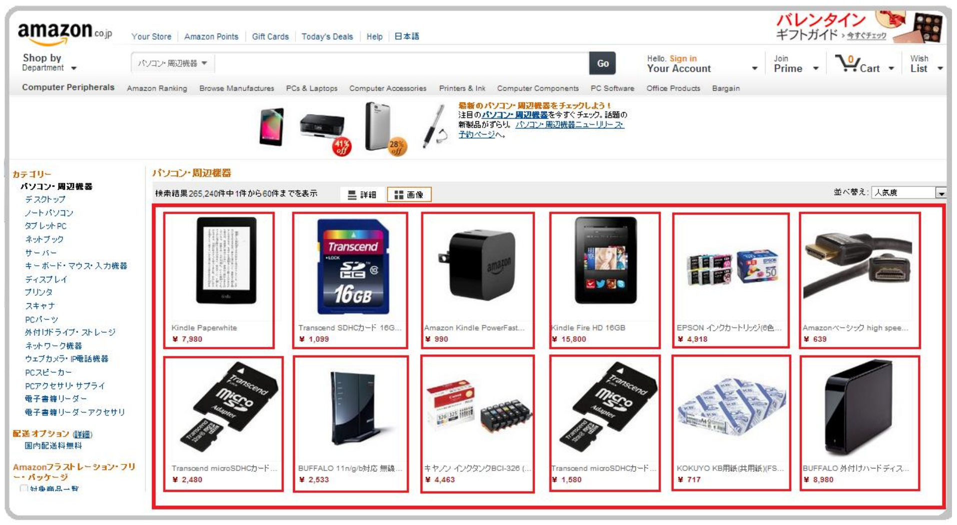


Figure 1. The informative blocks in a deep web page

However, no matter how the HTML code changes, the visual features of the informative blocks remain unchanged. An informative block of a deep web page is defined as a logical part that contains its core content. Example of informative blocks is shown in Figure 1. In a shopping mall page, the informative blocks are the product records. These informative blocks of deep web page always have a similar layout. For example in Figure 1, all the product records have the same layout: a product picture at the top, a product name in the middle, and a price at the bottom. Moreover, the informative blocks are the most significant parts of a deep web page. The web designers always attempt to make them as attractive as possible. Thus the informative blocks always occupy the largest area in the deep web pages.

Based on this motivation, this paper proposes a method to extract the informative blocks from a deep web. In this paper, we consider the rectangular region of an HTML element in a page as a visual block (see Section 3 for the detail). We transform the element layout of a visual block into a layout tree. By calculating the similarity of layout trees, we cluster the visual blocks that have the similar layout feature. Finally, the cluster which has the largest area is extracted as the informative block cluster. Our method is record-level, single-page and single data region based Web information extraction method.

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. The generation and similarity of layout tree are introduced in Section 4. Our solution for extract informative blocks is described in Section 5. Experimental results are reported in Section 6. Finally, conclusion and future work are given in Section 7.

2. Related Work

In the past few years, many approaches to information extraction have been proposed. Good surveys of previous works on Web information extraction can be found in [2] and [3].

One widely adopted technique to automatically detect, segment and extract data records is to search for repetitive patterns in HTML source code by calculating the similarity of DOM tree, tag tree or tag path. C. Castillo et al. [4] 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 Web pages or collections of interconnected Web pages. X. Ji et al. [5] 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. Jian et al. [6] proposed a Web information extraction method using the improved maximum entropy algorithm. They parsed the webpage to construct DOM tree, and then built pattern tree filtering the spam information. Finally, thet pruned the nodes with larger information maximum entropy and output the extracted theme information page. However, these methods have a number of limitations. For example, there might be cases where two visual blocks look exactly the same but they are coded differently, for instance one can code a block with a DIV element or with a TABLE element.

Recently, a few proposals for extracting Web information by using visual information have been reported in literature. J. Kang et al. [7] proposed an informative block extraction approach. This approach relies on visual clues 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 the tree edit distance method. Then it recognizes the informative block cluster by using tree alignment and tree matching. W. Liu et al. [8] proposed a vision-based IE method that primarily utilizes the visual features on deep Web pages to implement deep Web data extraction, including data record extraction and data item extraction. However if there are some unloaded images or missing style information, these methods may fail to build correct a visual containment tree which leads to data segmentation and extraction problems.

Some other methods that make use of ontology are proposed [12]. These methods also mainly analyze the HTML source and the ontology only plays a supplementary role.

3. Blocks with Similar Layout feature

3.1. Definition of Visual Block

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 2.

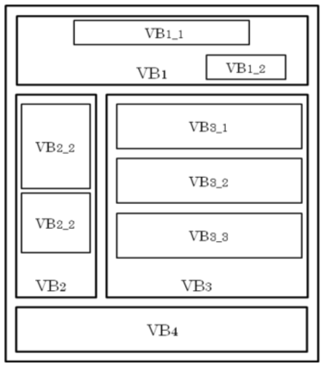


Figure 2. The visual blocks in a Web page

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 [9]. 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. The Layout of Visual Blocks

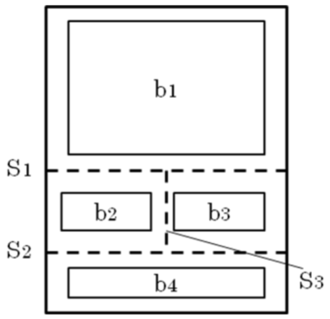


Figure 3. The layout of a visual block

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, SP). $LB = \{b_i \mid b_i : leaf\ and\ b \subset B, i \in [1, n]\}$ 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. $SP = \{S_1, S_2, \ldots, S_{n-1}\}$ 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.

4. The Layout Tree of Visual Blocks

4.1. Generation of 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

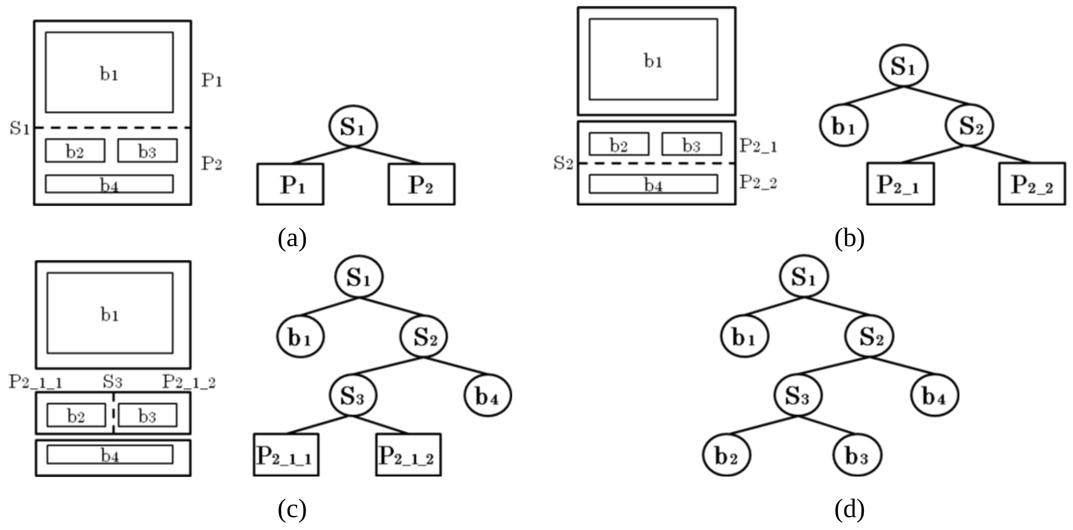


Figure 4. The generation of layout tree generation

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_{21_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.

4.2. 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) :

$$\text{Weight}(b_i) = \frac{\text{Area}(b_i)}{\text{Area}(B)} \tag{1}$$

Where B is the visual block where bi is in, Area(bi) and Area(B) represent the area of bi and B. In other words, in a 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(Si) of a separator Si is calculated as formula (2):

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

Where 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*.

4.3. Similarity of Layout Tree

As mentioned above, 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 [10]. 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 [10] 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:

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

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

Where 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)

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

Where n_1 *sim* n_2 represents n_1 and n_2 are the same, and n_1 *diff* n_2 represents n_1 and n_2 are different nodes. 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 the same 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 *sim* 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):

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

Where 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:

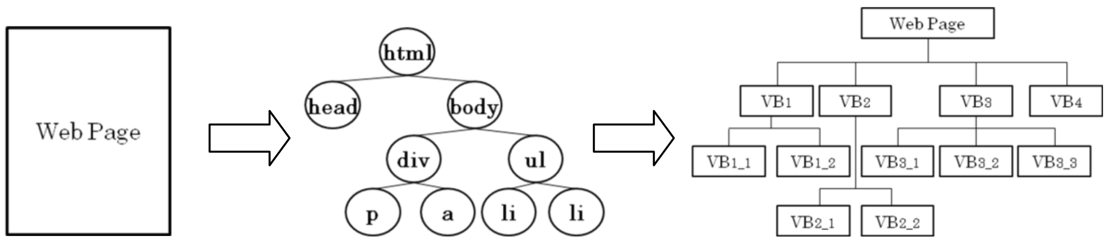
(1) $\mathrm{Sim}(T_1,T_2)\in[0,1]$

(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 α. If Sim(T_1, T_2) $\le \alpha$, then T_1 and T_2 are similar, otherwise they are different.

5. Extraction of Informative Blocks

5.1. Generation of Visual Block Tree

In order to generate the layout trees, the visual block of each HTML element needs to be determined. As shown in Figure 1, a Web page is made up of finite visual blocks. According to Definition III-1, each visual block has a corresponding DOM Element object. Thus, we first obtain the DOM tree of the Web page. By analyzing the properties of the DOM Element objects, the corresponding rectangular region can be determined. It should be noted the Element object does not contain the absolute coordinate of the corresponding HTML element. It only contains a relative



(a) Web page (b) DOM tree (c) Visual block tree

Figure 5. Generation of a visual block tree

coordinate to the parent HTML element. Fortunately, some browsers provide APIs to get absolute coordinate easily. After both the Element objects and rectangular regions are determined, the visual blocks are also determined.

According to the inclusion relation of visual blocks that is defined by Definition III-2, a DOM tree can be transform into a Visual Block Tree. Figure 5 shows the process for generating a visual block tree and Figure 5(c) is the visual block tree of the example in Figure 1. The visual block tree is an ordered tree. The blocks with the same parent have the same order as the corresponding HTML elements. Compared with a DOM tree, visual block tree removes the HTML elements that are not visible. Thus the visual block tree can be viewed as a pruned tree of a DOM tree. After the visual block tree is generated, for a given visual block *VB*, all the leaf blocks of *VB* can be determined and the layout tree of *VB* can also be generated.

5.2. Extraction of Informative Blocks

After the visual block tree is generated, we calculate the similarity of two blocks that are at the same depth using layout tree. If their similarity is less than the threshold, they will be clustered into a group. The detailed algorithm is shown as in Figure 6. These clusters contain both the informative blocks and meaningless blocks. We extract the informative blocks in the following steps:

STEP 1: Given two clusters $C_1 = \{VB_1,\ldots VB_i,\ldots VB_n\}$ and $C_2 = \{VB_1,\ldots VB_j,\ldots VB_m\}$, if $\forall\ VB_i \in C_1$, $\exists\, VB_j \in C_2$, and $VB_i \subset VB_j$, then we remove C_2.

STEP 2: Given two clusters $C_1 = \{VB_1,\ldots VB_i,\ldots VB_n\}$ and $C_2 = \{VB_1,\ldots VB_j,\ldots VB_m\}$, if $\exists\, VB_i \in C_1$, $VB_j \in C_2$, and VB_i and VB_j are layout similar, then we combine C_1 and C_2.

STEP 3: We calculate the area of each cluster as in formula (7):

$$\mathrm{Area}(C) = \sum \mathrm{Area}(VB_i) \quad (VB_i \in C) \tag{7}$$

Where Area(VB_i) presents the area of VB_i. Then we sort the clusters by area. The cluster that has the biggest area is extracted as the informative cluster. The blocks in the informative cluster are the informative blocks. Because the informative blocks are the most significant parts of a deep Web page, they always occupy the largest area in deep Web pages.

```
Input: a visual block tree T
Output: set of similar layout block clusters clusterSet

Begin
clusterSet = null                                  // initialize the set of layout similar block cluster
d = Depth(T)  //get the depth of T
loop i = 1 : d
   NodeList = getNodes(T, i)                       //get the nodes in depth i of T, and put into the array NodeList
   loop n = 1 : Length(NodeList) -1                // Length(Nodes) returns the length of the array NodeList
       loop m = n + 1 : Length(NodeList)
           if NodeList[n] or NodeList[m] is a leaf block          //if the leaf blocks do not include other blocks,
               then continue                                      // we do not consider their layout
           if NodeList[n] and NodeList[m] are already in cluster  //they may in a same cluster or different clusters
               then continue
           if  NodeList[n] and NodeList[m] are similar
               if NodeList[n] is already in a cluster C
                   then push( NodeList[m], C)
               if NodeList[m] is already in a cluster C
                   then push( NodeList[n], C)
               if both the NodeList[n] and NodeList[m] are not in cluster
                   then create a new cluster C'
                   push(NodeList[n], C')
                   push(NodeList[m], C')
                   push (C', clusterSet)                  //push cluster C' into clusterSet
       loop end
   loop end
loop end
return clusterSet
End
```

Figure 6. The algorithm to extract the layout similar blocks

6. Experiment and Evaluation

In this section, an experiment is conducted. The goal of the experiment is to determine whether the proposed method can extract informative blocks accurately and effectively. Moreover, we attempt to figure out an optimal threshold α.

6.1. Data Set and Method

We collected data from 10 different Web sites in order to guarantee the diversity of the data set. The 10 Web sites can be roughly divided into four types: online shopping sites (Amazon, Rakuten, Kakaku), news sites (Google news, Yahoo news, Goo news), video sites (YouTube, MSN Video), SNS and blog sites (Twitter, Laplog). For each site, we submitted 10 queries and collected one search result page for each query. Finally, we collected 10 * 10 = 100 pages as the data set. In these pages, the search result blocks are the informative blocks.

We let the threshold α be 0.1 to 1, and extract the informative blocks from the collected pages using proposed method. This method always selects the block cluster that has largest area as the informative cluster. We consider the areas of blocks is an important factor. Thus we propose a novel formulation of the precision, recall and F-measure. Given a Web page P and threshold α_i, the total area of extracted blocks is denoted TotalArea(P, α_i), in which the area of extracted informative blocks is denoted InfoArea(P, α_i). The precision, recall and F-measure are defined as in formula (8):

$$\mathrm{Precision}(P,\alpha_i)=\frac{\mathrm{InfoArea}(P,\alpha_i)}{\mathrm{TotalArea}(P,\alpha_i)}$$
$$\mathrm{Recall}(P,\alpha_i)=\frac{\mathrm{InfoArea}(P,\alpha_i)}{Max\{\mathrm{InfoArea}(P,\alpha_1),\ldots\mathrm{InfoArea}(P,\alpha_{10})\}}$$
$$\mathrm{F}(P,\alpha_i)=\frac{2\times\mathrm{Precision}(P,\alpha_i)\times\mathrm{Recall}(P,\alpha_i)}{\mathrm{Precision}(P,\alpha_i)+\mathrm{Recall}(P,\alpha_i)} \quad (8)$$

After the precision, recall, and F-Measure of each page is determined, we calculate the average value Precision(α_i), Recall(α_i), and F(α_i) of 100 pages.

6.2. Result and Analysis

Table 1. Result of Experiment

α_i	*Precision(α_i)*	*Recall(α_i)*	*F(α_i)*
0.1	0.9726	0.8299	0.8957
0.2	0.9874	0.8725	0.9264
0.3	0.9900	0.8942	0.9396
0.4	1.0000	0.9783	0.9890
0.5	0.9747	0.9838	0.9792
0.6	0.9742	0.9724	0.9735
0.7	0.9590	0.9827	0.9707
0.8	0.9559	0.9835	0.9695
0.9	0.9369	0.9812	0.9585
1.0	0.9287	0.9961	0.9612

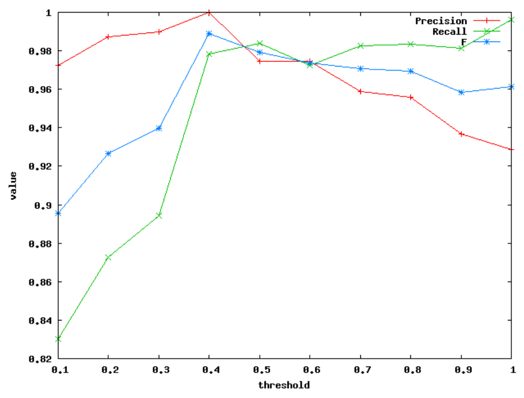


Figure 7. The diagram of result

Table 1 shows the experiment results and Figure 7 is a diagram illustrating the results. The results show that the proposed method is effective to extract informative blocks. The Precision(α_i) and F(α_i) reach a maximum when α=0.4. It also shows that the layout tree method can identify layout similar blocks accurately and effectively.

It should be noted that the change of the curves is different, when α is in different intervals. When α is in the interval [0.1, 0.4], the precision increases gradually. Compared with Precision(0.1), Precision(0.4) only changed 2.8%. While α is in the interval [0.4, 1], the precision decreases steeply. Compared with Precision(0.4), Precision(1.0) changed 7.1%. In other words, the ability of layout tree to eliminate layout different blocks decreases steeply when α is greater than 0.4. Conversely, when α is in the interval [0.1, 0.4], the recall increases steeply. Compared with Recall(0.1), *Recall(0.4)* changed 17.9%. While α is in the interval [0.4, 1], the recall changes gradually. Compared with Recall(0.4), Recall(1.0) only changed 1.8%. It means that when α is less than 0.4, the ability of layout tree to recognize the layout similar blocks significantly improves while α increases. If α is greater than 0.4, the ability of layout tree to recognize layout similar blocks improves gradually. Therefore, when α = 0.4, the layout tree is optimal for recognizing layout similar blocks from other blocks.

7. Conclusion and Future Work

Due to the explosive growth and popularity of the deep Web, information extraction from deep Web page has gained more and more attention. However, the HTML structure of Web documents has become more complicated, making it more difficult to recognize the target content by only analyzing the HTML source code. In this paper, we proposed a method to extract the informative blocks from a deep Web using the layout feature. We consider the rectangular region of an HTML element in a page as a visual block. We transformed the elements' layout of a visual block into a layout tree. By calculating the similarity of layout trees, we clustered the visual blocks that have similar layout features. Finally, the cluster which has the largest area was extracted as the informative block cluster. An experiment was conducted to determine whether the proposed method can extract informative blocks accurately and effectively. The experiment results showed that the proposed method is effective for extracting informative blocks from deep Web pages. Moreover, the experiment results also showed that this method is optimal when the threshold of layout tree similarity is 0.4.

However, the proposed method can only process deep Web pages containing one data region, while there is significant number of multidata-region deep Web pages. Though Zhao et al. [11] have attempted to address this problem. Their solution is HTML-dependent and its performance leaves large improvement for further. We intend to apply a layout-tree-based approach to tackle this problem in the future.

7. References

[1] Bergman, Michael K., "The Deep Web: Surfacing Hidden Value". The Journal of Electronic Publishing, Vol. 7, Issue 1, 2001.

[2] Chia-Hui Chang, Mohammed Kayed, Moheb Ramzy Girgis, and Khaled Shaalan, "A Survey of Web Information Extraction Systems", IEEE Transactions on Knowledge and Data Engineering, Vol. 18, Issue 10, pp. 1411-1428, 2006.

[3] Alberto H. F. Laender, Berthier A. Ribeiro-Neto, Altigran S. da Silva, and Juliana S. Teixeira, "A Brief Survey of Web Data Extraction Tools," SIGMOD Record, vol. 31, no. 2, pp. 84-93, 2002.

[4] Carlos Castillo, H´ector Valero, Jos´e Guadalupe Ramos, and Josep Silva, "Information extraction from webpages based on DOM distances", Lecture Notes in Computer Science, Vol. 7182, pp. 181-193, 2012.

[5] Xiangwen Ji, Jianping Zeng, Shiyong Zhang, and Chengrong Wu, "Tag tree template for Web information and schema extraction", Expert Systems with Applications, Vol. 37, Issue 12, pp. 8492–8498, 2010.

[6] Xue Jian, "Research on Web Information Extraction Based on the Improved Maximum Entropy Algorithm", AISS, Vol. 4, No. 13, pp. 85 ~ 91, 2012.

[7] Jinbeom Kang and Joongmin 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.

[8] Wei Liu, Xiaofeng Meng, and Weiyi 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.

[9] http://www.w3.org/

[10] Kaizhong Zhang and Dennis Shasha, "Simple Fast Algorithms for the Editing Distance between Trees and Related Problems", SIAM J. COMPUT, Vol. 18, No, 6, pp. 1245-1262, 1989.

[11] Hongkun Zhao, Weiyi Meng, and Clement Yu, "Automatic Extraction of Dynamic Record Sections from Search Engine Result Pages", In Proceedings of the VLDB '06, pp. 989-1000, 2006.

[12] Wu Hengliang, Zhang Weiwei, "A Web Information Extraction Method Based on Ontology", AISS, Vol. 4, No. 8, pp. 199 ~ 206, 2012