Web Page Segmentation with Structured Prediction and its Application in Web Page Classification *

Lidong Bing† Rui Guo♮ Wai Lam† Zheng-Yu Niu♮ Haifeng Wang♮
†Key Lab of High Confidence Software Techs., Ministry of Education (CUHK Sub-Lab), Hong Kong
†Dept. of Systems Engg. & Engg. Management, The Chinese University of Hong Kong, Hong Kong
♮Baidu Inc., Beijing, China
ldbing@se.cuhk.edu.hk, wlam@se.cuhk.edu.hk
grxkwok@gmail.com, niuzhengyu@baidu.com, wanghaifeng@baidu.com

ABSTRACT

We propose a framework which can perform Web page segmentation with a structured prediction approach. It formulates the segmentation task as a structured labeling problem on a transformed Web page segmentation graph (WPS-graph). WPS-graph models the candidate segmentation boundaries of a page and the dependency relation among the adjacent segmentation boundaries. Each labeling scheme on the WPS-graph corresponds to a possible segmentation of the page. The task of finding the optimal labeling of the WPS-graph is transformed into a binary Integer Linear Programming problem, which considers the entire WPS-graph as a whole to conduct structured prediction. A learning algorithm based on the structured output Support Vector Machine framework is developed to determine the feature weights, which is capable to consider the inter-dependency among candidate segmentation boundaries. Furthermore, we investigate its efficacy in supporting the development of automatic Web page classification.

Categories and Subject Descriptors

H.3 [**Information Storage and Retrieval**]: Information Search and Retrieval

Keywords

Web page segmentation, Web page classification, structured prediction, integer linear programming

* The work described in this paper is substantially supported by grants from the Research Grant Council of the Hong Kong Special Administrative Region, China (Project Code: CUHK413510) and the Direct Grant of the Faculty of Engineering, CUHK (Project Code: 4055034). This work is also affiliated with the CUHK MoE-Microsoft Key Laboratory of Human-centric Computing and Interface Technologies.
The work was partially done when the first author visited Baidu. We thank Dr. Baiyi Wu for discussing the optimization problem.

SIGIR'14, July 6–11, 2014, Gold Coast, Queensland, Australia.

http://dx.doi.org/10.1145/2600428.2609630.

1. INTRODUCTION

Web pages are typically designed to facilitate visual interaction with the human readers. The designer normally organizes the information of a page into different units or functional types [22], which are arranged in coherent visual segments in the page, such as header, footer, navigation menu, major content, etc. It is a trivial step to recognize those visual segments for readers. However, it is still a challenging problem for computer since the source code of Web pages is not encoded in such a way to differentiate the semantic blocks. To overcome the gap between the manner of the pages designed/read by human and the manner of the pages operated by the computer, Web page segmentation is regarded as an essential task in Web information mining. The aim of Web page segmentation is to decompose a Web page into sections that reveal the information presentation logic of the page designer and appear coherent to the readers. For example, if we consider the page in Figure 1, Web page segmentation should recognize those segments separated by the red boundary lines. Identifying such segments is useful for different downstream applications. One application is to re-organize a Web page so that it can be properly displayed or redecorated for devices with small-sized screen [1, 17, 20]. Then, people with visual impairment can easily digest them with their screen readers [21]. Link analysis, Web document indexing, and pseudo-relevance feedback can be more effective with the appropriate segments detected [12, 22, 29]. Duplicate content detection, Web page classification, and content change detection can be conducted on finer information units so as to obtain better performance [7, 19, 27].

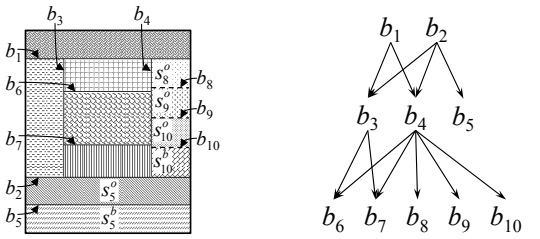
One group of previous works on Web page segmentation is heuristics-based [6, 19]. Cai et al. employed heuristic rules to capture structure features and visual properties [6]. Their method recursively segments the larger blocks into smaller ones in a top-down manner. However, this greedy manner is myopic and it may be trapped locally and stop to search better solutions. In addition, the patterns used in page design are unlimited and cannot be covered by a finite set of rules. Kohlschütter and Nejdl employed text density in different portions of a Web page as a clue to conduct segmentation [19]. Their method ignores other types of information such as image, frames and whitespace which provide useful clues for page segmentation. Chakrabarti et al. proposed a graph-theoretic approach to deal with Web page segmentation [7]. They cast the problem as a minimum cut problem on a weighted graph with the nodes as the DOM tree nodes and the edge weights as the cost of placing the end nodes in

Figure 1: An example of Web page segmentation.

the same segment or different segments. A learning based method was developed to determine the weights of edges. This method sometimes reports non-rectangular segments which are generally inaccurate. Different from [6], both [19] and [7] only obtain a flat segmentation of a Web page without knowing the hierarchical structure of the segmentation.

In this paper, we propose a framework to solve the above shortcomings of the existing works. Our framework performs Web page segmentation with a structured prediction approach by formulating the segmentation task as a structured labeling problem on a transformed Web page segmentation graph (WPS-graph). WPS-graph is a directed acyclic graph, as exemplified in Figure 2(b), and its vertices are composed of the candidate segmentation boundaries of the corresponding page, as depicted in Figure 2(a). Each vertex, i.e. boundary, is able to split the current segment into two sub-segments. The directed edges capture the dependency relation between the associated boundaries. Each labeling scheme of the WPS-graph, which assigns a binary label for each vertex to indicate whether or not the boundary should split the current segment, corresponds to a possible segmentation of the page. Different from the heuristic rule-based top-down search in [6], our framework performs the label prediction simultaneously for all vertices of the WPS-graph with a trained statistical model. The hierarchical structure of the intermediate segmentation steps is recorded by the layers in the WPS-graph. Thus, the output of our framework provides a good structural analysis of pages enabling better utilization for different Web mining tasks such as page classification. Furthermore, as exemplified with Figure 2, our framework can automatically avoid non-rectangular segments via the horizontal and vertical boundaries.

DOM structure, visual property, and text content features of a Web page are jointly considered in our framework. DOM structure features capture the structure characteristics of the segments, such as the structure similarity of the neighboring segments, the regularity of the DOM structure, etc. Visual property features capture visual clues of segments, such as background color, whitespace, font size, etc. The text content features capture some important semantic characteristics of the segments, such as the text similarity of the neighboring segments, the title keywords in a segment, etc. To allow different impacts of the features, a weight is associated with each feature. A machine learning algorithm based on the structured output Support Vector Machine framework [32] is developed to determine the feature weights.



(a) An example page. (b) WPS-graph of (a).

Figure 2: An example page and its WPS-graph.

The learning algorithm can consider the inter-dependency among the vertices of a WPS-graph during the training process. Therefore, the feature weights are determined with a global view on WPS-graphs but not merely on individual segmentation boundaries. To infer the optimal labeling, we consider the entire WPS-graph as a whole to conduct a structured prediction in which the labeling determination of one vertex, i.e., boundary, is coordinated with the others. To do so, we transform the labeling task into a binary Integer Linear Programming (ILP) problem [35], whose linear programming relaxation can be efficiently solved with the simplex algorithm [11].

Extensive experiments have been conducted on a large data set. The results demonstrate that our framework achieves better performance compared with two state-of-the-art methods. To investigate the efficacy of our framework in supporting other Web mining tasks, another experiment, namely Web page classification, is conducted. In this task, we propose a novel method to assemble the segmentation output of a page for constructing a more effective feature vector.

2. RELATED WORK

Web page structure analysis has been one hot research area for the past decade. There are several directions in this area, including single page oriented segmentation [6, 7, 19], site-oriented page segmentation [14], informative content extraction [13, 25, 31, 34], template (or boilerplate) detection [2, 33, 37], data record detection and entity extraction [3, 4, 5, 16, 24], etc. These directions are closely interwoven. Some directions share similar methodologies but aim at different products, such as site-oriented page segmentation and template detection. Some can be regarded as a preparation step for another purpose.

Single page oriented segmentation aims at segmenting an input Web page into distinct coherent segments or blocks (such as main content, navigation bars, etc.) based on its own content. Besides the works [6, 7, 19] discussed above, Chen et al. [9] distinguish five block types, namely, header, footer, left side bar, right side bar, and main content. Fernandes et al. [14] proposed a site-oriented method for Web page segmentation. Hattori et al. proposed a segmentation method based on calculating the distance between content elements within the HTML tag hierarchy [17].

Aiming at improving the performance of Web page clustering and classification, Yi et al. proposed a site style tree (SST) structure that labels DOM nodes with similar styles across pages as uninformative [37]. In [15], the URL features were shown to be effective in homepage identification, which can be regarded as a special case of page functional type classification.

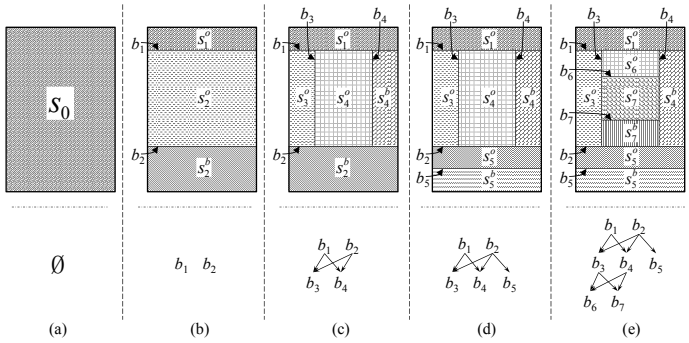


Figure 3: Steps of WPS-graph construction.

Algorithm 1: Construction of WPS-graph.

```
1: initialization: s_0 ← p, Q.enqueue(s_0), G ← ∅
2: while ¬Q.empty() do
3:    s ← Q.dequeue()
4:    if s is separable then
5:       add the boundaries b_{i..j} in s as vertices into G
6:       add new edges for b_{i..j}
7:       Q.enqueue(subsegments of s)
8:    end if
9: end while
```

3. PROBLEM FORMULATION

3.1 Web Page Segmentation

Web page segmentation is the task of breaking a page into sections that reveal the information presentation structure of the page designer and appear coherent to the readers. To facilitate the description of our framework, one illustration page example is given in Figure 2(a). The lines in the page are candidate boundaries for splitting the page into visual segments. The solid lines are the boundaries that should split the segments where the boundaries locate into sub-segments. For example, b_1 and b_2 split the entire page into the upper, middle, and lower parts. While the dashed lines are the boundaries that do not perform splitting operation, such as b_8, b_9, and b_{10}. Therefore, the final segmentation result is as depicted by the solid boundaries. Performing page segmentation is equivalent to determining a label assignment which assigns the label `Y` (splitting operation) or the label `N` (not a splitting operation) to each boundary. For the page in Figure 2(a), the solid boundaries take the label `Y`, while the dashed boundaries take the label `N`. A segmentation should follow the structure constraints of the page. For the above example, only after the boundaries b_1, b_2, b_3 and b_4 simultaneously decide to split, it becomes meaningful to consider which label b_6 and b_7 should have. If any one of b_1, b_2, b_3 and b_4 is labeled with `N`, b_6 and b_7 are automatically labeled with `N`. This kind of constraints compose of a topology graph $\mathcal{G}$, named Web page segmentation graph (WPS-graph) in this paper. The WPS-graph of the page in Figure 2(a) is given in Figure 2(b).

Let $\mathbf{C}$ denote a label assignment by taking a label from {`Y`, `N`} for each vertex, i.e., boundary, in $\mathcal{G}$. The segmentation task is formulated as the following optimization problem:

$$\mathbf{C}^* = \underset{\mathbf{C}}{\operatorname{argmax}} F(\mathcal{G}, \mathbf{C}; \mathbf{w}), \tag{1}$$

where F is an objective function that evaluates the fitness of $\mathbf{C}$ for $\mathcal{G}$. The variable $\mathbf{w}$ gives the weights of the DOM structure, visual property and text content features. Such design globally evaluates the fitness of a label assignment $\mathbf{C}$ for $\mathcal{G}$ so that the determined segmentation is more accurate. Note that each legal label assignment must satisfy all the dependency constraints depicted by the edges of $\mathcal{G}$. Fundamentally, some existing methods such as [6, 7, 19] can also be represented in the form of $F(\mathcal{G}, \mathbf{C}; \mathbf{w})$. For example, the visual block tree approach in VIPS [6] can be transformed into our WPS-graph. And recursively segmenting the larger blocks can be transformed into determining the labels of the corresponding boundaries. The DoC criteria in VIPS can be regarded as simple DOM structure and visual property features. Different from these methods, our model conducts a global evaluation on the fitness of a segmentation for a page.

3.2 WPS-graph

Definition 1 (WPS-graph). *For a Web page p, its WPS-graph $\mathcal{G} = \{\mathbf{B}, \mathbf{E}\}$ is an acyclic directed graph. Each vertex b in $\mathbf{B}$ is a candidate segmentation boundary in p. Each directed edge $e : \langle b_i, b_j \rangle$ in $\mathbf{E}$ indicates the constraint $(\mathbf{C}(b_j) = \texttt{Y}) \longrightarrow (\mathbf{C}(b_i) = \texttt{Y})$, where $\mathbf{C}$ is a label assignment for the vertex set $\mathbf{B}$ of $\mathcal{G}$.*

The construction of WPS-graph for a page is described in Algorithm 1. Initially, the entire page is regarded as one segment and denoted as s_0. Meanwhile, the WPS-graph is $\emptyset$. Refer to Line 1 in Algorithm 1 and part (a) in Figure 3. While the segment queue Q is not empty, the segment at the front of Q is processed, as depicted in Line 3. Take s_0 as an example as depicted in part (b) in Figure 3. The candidate horizontal boundaries b_1 and b_2 split s_0 into three sub-segments and they are added as vertices into $\mathcal{G}$, as given in Line 5 in Algorithm 1. For the new vertices, namely, b_1 and b_2, no new edges need to be added since s_0 is the entire page and b_1 and b_2 do not depend on any previous boundary. The current $\mathcal{G}$ is as depicted in the lower section of part (b) in Figure 3. The sub-segments, namely, s_1^o (on the boundary b_1), s_2^o (also known as s_1^b), and s_2^b (beneath the boundary b_2) are enqueued, as shown in Line 7 in Algorithm 1. To continue, after s_1^o is found inseparable, s_2^o is processed as depicted in part (c) in Figure 3. The boundaries b_3 and b_4 are added as new vertices in $\mathcal{G}$. b_3 and b_4 depend on b_1 and b_2 so that the edges are added accordingly as depicted in the lower section of part (c). The sub-segments, namely, s_3^o, s_4^o and s_4^b, in the upper section of part (c) are enqueued. For the sake of simplicity, we also use s_i^o to denote the sub-segment on the left of b_i and s_i^b to denote the sub-segment on the right of b_i. Then, s_2^b is processed as depicted in part (d) in Figure 3. The boundary b_5 and the edge $\langle b_2, b_5 \rangle$ are added into the graph. The construction process terminates until Q is empty.

Several issues should be noted in the construction. First, a candidate boundary cannot cut across any sub DOM tree. For example, there is no subtree whose one part is in s_1^o and the other part is in s_2^o. Second, when judging the separability of a segment, the horizontal boundaries inside it are examined first since they are more commonly used than vertical boundaries. Third, in separability judgment, if a segment is composed of a single DOM tree, we recursively use the lower level subtrees of it instead. For example, if s_0 is composed of a single <table> and the table has several <tr>'s, we

will find the boundaries between each neighboring pair of <tr>'s. If s_2^o is composed of a single <table> with a single <tr>, we will find the boundaries between each neighboring pair of <td>'s of the <tr>. Fourth, each vertex at most directly depends on two other vertices. Theoretically, a candidate boundary b inside the segment s depends on all four boundaries of s, among which b indirectly depends on at least two and at most three of them. In part (e) in Figure 3, we can see that b_6 and b_7 indirectly depend on b_1 and b_2.

3.3 Informative Boundary

The segments with essential information are normally arranged in conspicuous position of a page, such as the middle of the first screen. Such segments are known as informative segment [13, 23]. Incorrectly segmenting informative segments causes larger loss. Taking the news content segment in a news page as an example, any segmentation that mistakenly segments the paragraphs of the content into different segments is not a favorable result. This is called over-segmentation mistake. On the other hand, if the news content segment is combined with other segments and becomes a subpart of the combined segment, insufficient-segmentation mistake occurs. To make the informative segments accurately segmented, we define the related boundaries as informative boundaries and give them some special treatment.

Definition 2 (Informative Boundary). *If a boundary is mistakenly labeled, the related informative segment will be overly or insufficiently segmented. Such boundary is an informative boundary.*

Suppose s_4^o is an informative segment in the demo page in Figure 3, b_1, b_2, b_3, b_4, b_6, and b_7 are informative boundaries. Since b_3 and b_4 depend on b_1 and b_2, we only need to ensure that b_3 and b_4 are correctly labeled as `Y`. Similarly, after b_6 and b_7 are correctly labeled as `N`, the boundaries that depend on b_6 and b_7 will be labeled as `N` automatically. Therefore, we define proper informative boundary as follows.

Definition 3 (Proper Informative Boundary). *Suppose b_i is an informative boundary and outside the informative segment, if there is no b_j which directly depends on b_i and is also outside the informative segment, b_i is a proper informative boundary. Suppose b_k is an informative boundary and inside the informative segment, if b_k directly depends on a proper informative boundary outside the informative segment, b_k is a proper informative boundary.*

In the above example, b_3, b_4, b_6, and b_7 are the proper informative boundaries.

4. FEATURES

Let $\mathbf{\Psi}(\mathcal{G}, \mathbf{C})$ denote the combined feature representation of $\mathcal{G}$ and its label assignment $\mathbf{C}$. Thus, the objective function F in Equation 1 is formulated as:

$$F(\mathcal{G}, \mathbf{C}; \mathbf{w}) \equiv \langle \mathbf{\Psi}(\mathcal{G}, \mathbf{C}), \mathbf{w} \rangle, \tag{2}$$

which is the linear combination of the features in $\mathbf{\Psi}(\mathcal{G}, \mathbf{C})$ with their corresponding weights given in $\mathbf{w}$. For each boundary b in $\mathbf{B}$ of $\mathcal{G}$, we define a group of features from its surrounding segments to assist the determination of its label. For the example in Figure 2(a), if the segments s_5^o and s_5^b have different background colors, the label `Y` is probably more suitable than the label `N` for the boundary b_5. If one text segment contains very similar terms as in the title of a news page, we probably should not split the boundary beneath this segment since it will separate the title and the main content of the news. According to the sources, the features are categorized into two types, namely, local feature, and context feature,

4.1 Local Features

Local features of a boundary b_i are computed based on the characteristics of its surrounding sub-segments, i.e., s_i^o and s_i^b. We design two types of local features, namely, local segment features, and local segment relation features.

Local segment features are designed to capture the characteristics of a single segment s_i^o or s_i^b. Let $\mathbf{\Phi}(s_i^o)$ denote the feature vector related to s_i^o. The combined feature map of s_i^o and the label c_i of b_i is denoted as:

$$\mathbf{\Psi}^l(s_i^o, c_i) \equiv \mathbf{\Phi}(s_i^o) \otimes \mathbf{\Lambda}^c(c_i), \tag{3}$$

where $\otimes$ is the operator of tensor multiplication, $\mathbf{\Lambda}^c(c_i)$ is the canonical representation of the label c_i:

$$\mathbf{\Lambda}^c(c_i) \equiv (\delta(\mathtt{Y}, c_i), \delta(\mathtt{N}, c_i)), \tag{4}$$

where $\delta(\mathtt{Y}, c_i)$ is an indicator function and has the value 1 if $c_i = \mathtt{Y}$ and the value 0 otherwise. As revealed by Equation 3, each single feature is mapped to a dimension according to the label c_i. Similarly, the combined feature map of the segment s_i^b and the label c_i is denoted as:

$$\mathbf{\Psi}^l(s_i^b, c_i) \equiv \mathbf{\Phi}(s_i^b) \otimes \mathbf{\Lambda}^c(c_i). \tag{5}$$

Local segment features include basic features, geographic features, color features, content features, text appearance features, text richness features, tag richness features, font size features, etc. Some examples are the number of links in the segment, the background color of the segment, the number of terms in a segment that also appear in the page title, the token based text density over the segment size, etc.

Local segment relation features reveal the relation of the two segments. Let $\mathbf{\Phi}(s_i^o, s_i^b)$ represent the features summarized for capturing the relations between s_i^o and s_i^b. The combined feature map is denoted as:

$$\mathbf{\Psi}^l(s_i^o, s_i^b, c_i) \equiv \mathbf{\Phi}(s_i^o, s_i^b) \otimes \mathbf{\Lambda}^c(c_i). \tag{6}$$

Such features include height difference in the DOM, text length difference, color difference, size difference, font size difference, typeface difference, text similarity, etc.

4.2 Context Features

Human readers also consider the context information in identifying page segments. Take the page given in Figure 2(a) as an example. In addition to the local features from s_{10}^o and s_{10}^b, the sibling segments s_8^o and s_9^o also provide useful hints to determine the label of b_{10}. Suppose the DOM structures of these four segments are similar, it is very likely that they present four records of the same type of information. Thus, it is probably not preferred to split the boundary b_{10}. We design two types of context features, namely, context segment features, and context segment relation features, for each b_i to capture the characteristics of the segments in the sibling sequence. Context segment features include the average DOM structure similarity with the sibling segments, occurrence frequency based on DOM structure similarity, mean and standard deviation of occurrence

intervals, etc. Context segment relation features reveal the relation of the two segments' context features, such as the difference of occurrence frequency, occurrence characteristics of the forest of them, etc. The combined context feature maps are denoted as:

$$\mathbf{\Psi}^c(s_i^o, c_i) \equiv \mathbf{\Upsilon}(s_i^o; \mathbf{S}) \otimes \mathbf{\Lambda}^c(c_i), \quad (7)$$

$$\mathbf{\Psi}^c(s_i^b, c_i) \equiv \mathbf{\Upsilon}(s_i^b; \mathbf{S}) \otimes \mathbf{\Lambda}^c(c_i), \quad (8)$$

$$\mathbf{\Psi}^c(s_i^o, s_i^b, c_i) \equiv \mathbf{\Upsilon}(s_i^o, s_i^b; \mathbf{S}) \otimes \mathbf{\Lambda}^c(c_i), \quad (9)$$

where $\mathbf{S}$ is the sequence of the corresponding sibling segments, $\mathbf{\Upsilon}$ is the features extracted according to the characteristics of the segments in $\mathbf{S}$.

4.3 Aggregated Feature Representation

By aggregating the above features, the combined feature map of a boundary b_i and its label c_i is presented as:

$$\mathbf{\Psi}(b_i, c_i) \equiv \left(\begin{pmatrix} \mathbf{\Psi}^l(s_i^o, c_i)' \\ \mathbf{\Psi}^l(s_i^b, c_i)' \\ \mathbf{\Psi}^l(s_i^o, b_i^b, c_i)' \end{pmatrix}' \begin{pmatrix} \mathbf{\Psi}^c(s_i^o, c_i)' \\ \mathbf{\Psi}^c(s_i^b, c_i)' \\ \mathbf{\Psi}^c(s_i^o, s_i^b, c_i)' \end{pmatrix}' \right). \quad (10)$$

The combined feature representation of a WPS-graph $\mathcal{G}$ and its label assignment $\mathbf{C}$ is the combination of the feature map from each boundary:

$$\mathbf{\Psi}(\mathcal{G}, \mathbf{C}) \equiv \sum_{b_i} \mathbf{\Psi}(b_i, c_i). \quad (11)$$

As shown above, different features are combined and the difference of their impacts will be captured by the corresponding weights in $\mathbf{w}$.

5. INFERENCE OF SEGMENTATION

To infer a label assignment satisfying the dependency constraints, one strategy is to consider the vertices one by one in the topological order as depicted by the WPS-graph. Only when all the ancestors of a vertex are labeled with `Y`, we will evaluate `Y` and `N` for it. Otherwise, we assign the label `N` to this vertex. However, this manner is myopic and cannot achieve an optimal solution. To overcome this problem, we formulate the label inference on $\mathcal{G}$ as a binary Integer Linear Programming (ILP) problem with the label dependency constraints transferred as the constraints of the binary ILP. The source code of the inference is publicly available at `http://www.se.cuhk.edu.hk/~textmine/`.

Let F_i^* denote the partial objective value achieved by the vertex b_i with the label c_i^* in the optimal label assignment $\mathbf{C}^*$. F_i^* can be represented as:

$$F_i^* = (F_i - \overline{F}_i)x_i + \overline{F}_i, \quad (12)$$

where $x_i = \delta(\mathtt{Y}, c_i^*)$. F_i and $\overline{F}_i$ are calculated as follows:

$$F_i = \langle \mathbf{\Psi}(b_i, \mathtt{Y}), \mathbf{w} \rangle, \quad (13)$$

$$\overline{F}_i = \langle \mathbf{\Psi}(b_i, \mathtt{N}), \mathbf{w} \rangle. \quad (14)$$

If $x_i = 0$, we have $F_i^* = \overline{F}_i$ and $c_i^* = \mathtt{N}$. Otherwise, we have $F_i^* = F_i$ and $c_i^* = \mathtt{Y}$. Thus, the optimal value of F can be computed as $F^* = \sum_{i=1}^n F_i^*$, where $n = |\mathbf{B}|$. The task of finding the optimal label assignment as given in Equation 1 can be transformed as solving a binary ILP problem:

$$\begin{aligned} \max \sum_{i=1}^n F_i^* \quad &= \quad \max \Big(\sum_{i=1}^n (F_i - \overline{F}_i)x_i + \overline{F}_i\Big), \\ &s.t. \quad x_i \in \{0, 1\}, \\ &\quad\ \ \langle b_i, b_j \rangle \in \mathbf{E} \Rightarrow x_i \geq x_j. \end{aligned} \quad (15)$$

The second constraint ensures that if there is an edge from b_i to b_j in $\mathbf{E}$ of $\mathcal{G}$ and b_i has the label `N`, i.e., $x_i = 0$, b_j must also have the label `N`, i.e., $x_j = 0$; if b_j has the label `Y`, i.e., $x_j = 1$, b_i must also have the label `Y`, i.e., $x_i = 1$. After removing the unchanged term $\overline{F}_i$ in F_i^*, Formula 15 can be equivalently written as the following compacted form:

$$\max \mathbf{f}'\mathbf{x}, \quad s.t.\ \mathbf{x} \in \{0, 1\}^n \ \ and \ \ A\mathbf{x} \leq \{0\}^m, \quad (16)$$

where $\mathbf{f}'$ is the coefficient vector and $f_i = F_i - \overline{F}_i$, $A \in \mathbb{R}^{m \times n}$ is the constraint matrix, where $m = |\mathbf{E}|$. Each constraint $x_i \geq x_j$ corresponds to one row in A whose j-th element is $+1$, i-th element is -1 and other elements are 0.

However, binary ILP has been proved to be NP-hard [35]. To solve it, we first relax the binary ILP in Equation 16 to a linear programming (LP) problem which has efficient and widely used solvers such as simplex [11]. Then, it can be proved that the linear relaxation has an integral optimal solution, which is thus also the optimal solution of the original binary ILP problem. The constraint $\mathbf{x} \in \{0, 1\}^n$ is written as $\mathbf{x} \in \mathbb{Z}_+^n$ and $\mathbf{x} \leq \{1\}^n$. $\mathbf{x} \leq \{1\}^n$ and $A\mathbf{x} \leq \{0\}^m$ are jointly written as $B\mathbf{x} \leq \mathbf{b}$, where $B = \begin{pmatrix} A \\ I_n \end{pmatrix}$, $\mathbf{b} = \begin{pmatrix} \{0\}^m \\ \{1\}^n \end{pmatrix}$. Thus, we get an ILP problem:

$$\max \mathbf{f}'\mathbf{x}, \quad s.t.\ \mathbf{x} \in \mathbb{Z}_+^n \ \ and \ \ B\mathbf{x} \leq \mathbf{b}. \quad (17)$$

The linear relaxation of the problem in Formula 17 is:

$$\max \mathbf{f}'\mathbf{x}, \quad s.t.\ \mathbf{x} \in \mathbb{R}_+^n \ \ and \ \ B\mathbf{x} \leq \mathbf{b}. \quad (18)$$

One nice property of the LP problem in Formula 18 is that the constraint matrix B is totally unimodular [35], which can be proved straightforwardly and is omitted due to the tight space. This property guarantees that the LP problem has an integral optimal solution for any integer vector $\mathbf{b}$ for which it has a finite optimal value [35]. Therefore, the integral optimal solution of the LP in Formula 18 is also an optimal solution of the ILP in Formula 17. Obviously, it is also an optimal solution of the binary ILP in Formula 16, from which the ILP is transformed.

To speed up the inference algorithm, the leaf vertex b_i in $\mathcal{G}$ whose value satisfies $\overline{F}_i \geq F_i$ can be safely labeled as `N` and removed from the inference procedure. The reason is that the labeling of its ancestor vertices does not affect b_i. Obviously, this removal preprocessing is recursive until each of the remaining leaf vertex b_j has $\overline{F}_j < F_j$.

6. TRAINING

We develop a machine learning algorithm based on the structured output Support Vector Machine framework [32] to determine the feature weights. Our learning algorithm considers the inter-dependency among the vertices of a WPS-graph during the training process. Therefore, the feature weights are determined with a global view on WPS-graphs but not merely on individual segmentation boundaries. The source code of this learning framework is publicly available at `http://www.se.cuhk.edu.hk/~textmine/`.

6.1 Learning Framework

Let $\{(\mathcal{G}_i, \mathbf{C}_i)\}_{i=1}^N$ denote a set of training data instances. The quadratic program form of the SVM model with slack

re-scaled by the loss is:

$$\min_{\mathbf{w},\xi} \frac{1}{2}||\mathbf{w}||^2 + \frac{C}{N}\sum_{i=1}^{N}\xi_i$$
$$s.t. \quad \forall i, \forall \mathbf{C} \in \mathcal{Y} \setminus \mathbf{C}_i : \langle \delta\Psi_i(\mathbf{C}), \mathbf{w}\rangle \geq 1 - \frac{\xi_i}{\Delta(\mathbf{C}_i, \mathbf{C})}, \quad (19)$$

where $\xi_i \geq 0$ is the slack variable of $\mathcal{G}_i$, $C > 0$ is a trade-off constant of the two parts and takes value 1 in this paper, $\mathcal{Y}$ is the set of all possible label assignments of $\mathcal{G}_i$, $\langle \delta\Psi_i(\mathbf{C}), \mathbf{w}\rangle = F(\mathcal{G}_i, \mathbf{C}_i; \mathbf{w}) - F(\mathcal{G}_i, \mathbf{C}; \mathbf{w})$ is the margin between the objective values of $\mathbf{C}_i$ and $\mathbf{C}$, and $\Delta(\mathbf{C}_i, \mathbf{C})$ denotes the loss caused by the label assignment $\mathbf{C}$. Similarly, the quadratic program form of the SVM model with margin re-scaled by the loss is:

$$\min_{\mathbf{w},\xi} \frac{1}{2}||\mathbf{w}||^2 + \frac{C}{N}\sum_{i=1}^{N}\xi_i$$
$$s.t. \quad \forall i, \forall \mathbf{C} \in \mathcal{Y} \setminus \mathbf{C}_i : \langle \delta\Psi_i(\mathbf{C}), \mathbf{w}\rangle \geq \Delta(\mathbf{C}_i, \mathbf{C}) - \xi_i. \quad (20)$$

Tsochantaridis et al. proposed a cutting plane based algorithm to solve this optimization problem in its dual formulation [32]. It selects a subset of constraints from the exponentially large set $\mathcal{Y}$ to ensure a sufficiently accurate solution. The procedure of finding the feature weights is briefly summarized in Algorithm 2. S_i is the working set of selected constraints for the instance $\mathcal{G}_i$, α's are the Lagrange multipliers, and ε is the precision parameter. The algorithm proceeds by finding the most violated constraint for $\mathcal{G}_i$ involving $\hat{\mathbf{C}}$ (refer to Line 5). If the margin violation of this constraint exceeds the current ξ_i by more than ε (refer to Line 7), the working set S_i of $\mathcal{G}_i$ is updated. α's and $\mathbf{w}$ are also updated with the updated working set accordingly. We refer the reader to [32] for more details of the algorithm.

In the learning procedure as depicted in Algorithm 2, it is required to optimize the cost function in Line 4 for finding the most violated constraint corresponding to $\hat{\mathbf{C}}$:

$$\hat{\mathbf{C}} = \operatorname*{argmax}_{\mathbf{C}\in\mathcal{Y}} H(\mathbf{C}). \quad (21)$$

The upper and the lower forms of $H(\mathbf{C})$ in Line 4 correspond to the slack re-scaling and margin re-scaling definitions as given in Formulae 19 and 20 respectively.

6.2 Optimization for Slack Re-scaling

Recall that the missing of the informative segments of a page causes larger loss. Our slack re-scaling formulation as given in Formula 19 is able to take this into consideration with a loss function defined based on informative segments:

$$\Delta(\mathbf{C}_i, \mathbf{C}) \equiv \exp\{\frac{\sum_{b\in\mathbf{B}_{info}} \bar{\delta}(\mathbf{C}_i(b), \mathbf{C}(b))}{|\mathbf{B}_{info}|}\}, \quad (22)$$

where $\mathbf{B}_{info}$ denotes the set of proper informative boundaries, $\mathbf{C}(b)$ is the label of b in $\mathbf{C}$, $\bar{\delta}$ is an indicator function which takes the value 0 if $\mathbf{C}_i(b) = \mathbf{C}(b)$ and takes the value 1 otherwise. If $\mathbf{B}_{info} = \emptyset$, we set $\Delta(\mathbf{C}_i, \mathbf{C}) = 1$.

To optimize the following cost function:

$$H(\mathbf{C}) \equiv (1 - \langle \delta\Psi_i(\mathbf{C}), \mathbf{w}\rangle)\Delta(\mathbf{C}_i, \mathbf{C}), \quad (23)$$

we enumerate a subset of possible loss value levels as defined in Equation 22. The derivation of the optimal $\hat{\mathbf{C}}$ for Equation 23 is summarized as Proposition 1.

PROPOSITION 1. *Let $\mathbf{C}^*$ denote the label that achieves the optimal value for F and $\mathbf{B}^{\times}_{info}$ denote the proper informative boundaries wrongly labeled in $\mathbf{C}^*$. Let $\{\mathbf{B}'_{info}\}$ be all subsets of $\mathbf{B}_{info}$ having more elements than $\mathbf{B}^{\times}_{info}$, and let $\mathbf{C}'$ be the label assignment that achieves the largest F value when all boundaries in $\mathbf{B}'_{info}$ are wrongly labeled and all boundaries in $\mathbf{B}_{info} \setminus \mathbf{B}'_{info}$ are correctly labeled. The label $\hat{\mathbf{C}}$ maximizing $H(\mathbf{C})$ is from $\cup_{\mathbf{B}'_{info}}\{\mathbf{C}'\} \cup \{\mathbf{C}^*\}$.*

Algorithm 2: Finding feature weights via structured output SVM learning.

1: **initialization**: $\{(\mathcal{G}_i, \mathbf{C}_i)\}_{i=1}^N$, C, ε, $\forall i : S_i \leftarrow \emptyset$
2: **repeat**
3: **for** $i = 1, \cdots, N$ **do**
4: $H(\mathbf{C}) \equiv \begin{cases} (1 - \langle \delta\Psi_i(\mathbf{C}), \mathbf{w}\rangle)\Delta(\mathbf{C}_i, \mathbf{C}) \\ \Delta(\mathbf{C}_i, \mathbf{C}) - \langle \delta\Psi_i(\mathbf{C}), \mathbf{w}\rangle \end{cases}$
5: $\hat{\mathbf{C}} = \operatorname{argmax}_{\mathbf{C}} H(\mathbf{C})$
6: $\xi_i = \max\{0, \max_{\mathbf{C}\in S_i} H(\mathbf{C})\}$
7: **if** $H(\hat{\mathbf{C}}) > \xi_i + \varepsilon$ **then**
8: $S_i \leftarrow S_i \cup \{\hat{\mathbf{C}}\}$
9: update α's and $\mathbf{w}$ with $\cup_i S_i$
10: **end if**
11: **end for**
12: **until** no S_i has changed during iteration

The proof of Proposition 1 is straightforward and omitted due to the space limitation.

6.3 Optimization for Margin Re-scaling

The margin re-scaling formulation in Formula 20 is designed to perform a general segmentation of pages. For this design, we define a hamming distance based loss function:

$$\Delta(\mathbf{C}_i, \mathbf{C}) \equiv \sum_{b\in\mathbf{B}} \bar{\delta}(\mathbf{C}_i(b), \mathbf{C}(b)), \quad (24)$$

where each boundary is treated equally. To optimize the second cost function as given in Equation 25:

$$H(\mathbf{C}) \equiv \Delta(\mathbf{C}_i, \mathbf{C}) - \langle \delta\Psi_i(\mathbf{C}), \mathbf{w}\rangle, \quad (25)$$

it is equivalent to optimize:

$$H'(\mathbf{C}) \equiv \Delta(\mathbf{C}_i, \mathbf{C}) + F(\mathcal{G}_i, \mathbf{C}; \mathbf{w}). \quad (26)$$

Let $\hat{\mathbf{C}}'$ denote the label that achieves the largest value of H', and $\hat{H}'_i$ denote the partial value of H' achieved by the vertex b_i having the label $\hat{\mathbf{C}}'(b_i)$, denoted as $\hat{c}'_i$. $\hat{H}'_i$ can be represented as:

$$\hat{H}'_i = (H'_i - \overline{H}'_i)x_i + \overline{H}'_i, \quad (27)$$

where $x_i = \delta(\mathtt{Y}, \hat{c}'_i)$. H'_i and $\overline{H}'_i$ are calculated as follows:

$$H'_i = \delta(c_i, \mathtt{Y}) + \langle \Psi(b_i, \mathtt{Y}), \mathbf{w}\rangle, \quad (28)$$
$$\overline{H}'_i = \delta(c_i, \mathtt{N}) + \langle \Psi(b_i, \mathtt{N}), \mathbf{w}\rangle. \quad (29)$$

Referring to Equations 12, 13, and 14, the task of finding the label assignment $\hat{\mathbf{C}}'$ maximizing H' can also be solved in the same way as given in Section 5.

7. EXPERIMENTS

7.1 Experimental Setting

Data Preparation. We categorize the pages on the Web into 10 broad types as given in the first column of Table 1. 1,000 pages of the first 9 types were randomly picked from a

Table 1: Different types of pages on the Web and the number of each type in our data set.

Functional type (ID)	Description	Page #
Index (1)	The major part of the page is composed of links for the navigation purpose, such as the thread list of forums, the home page of a portal site, etc.	366
Image (2)	The page is designed to present images.	180
Forum (3)	Presenting the major content of forum posts.	100
Product (4)	Presenting the detailed descriptions of products.	71
Search Result (5)	Presenting the search result of various search engines.	70
Blog (6)	Presenting the major content of Blog posts.	69
Download (7)	Presenting the download information for softwares, music, and videos.	50
News (8)	Presenting the major content of the daily news, or the major content of various articles.	48
Video (9)	The page is designed to present videos.	46
Other (10)	Including adult pages, wap pages, etc.	0
Sum		1,000

large page repository, since adult pages are normally omitted by applications and wap pages are designed differently compared with normal Web pages. The number of pages in different types is given in the third column of Table 1. The total number of pages in our data set is almost 10 times of that used in some previous works [7, 19]. In addition, the smallest type contains 46 pages, which is a reasonable number for conducting type-specific experiment. To perform data annotation, we developed a browser-based user-friendly tool for annotators to specify the label of each boundary. After the annotators finish annotating one page, they label the informative segment in the page. If the page is not an index page, the segment that presents the major information of the page is annotated as informative segment, such as the news content segment of a news page and the result segment of a Web search page.

Evaluation Metrics. The segmentation result generated by a Web page segmentation method groups the visual elements of a Web page into cohesive regions visually and semantically. Similar to the previous works [7, 19], we regard each generated segment as a cluster of visual elements and employ cluster correlation metrics to conduct the evaluation. The first metric is the Adjusted Rand Index (ARI) [18]. Rand Index is defined to measure the agreement between an output clustering and the ground truth clustering by counting the pairs of elements on which two clusterings agree [28]. The Rand Index lies between 0 and 1, with 0 indicating that the two clusters do not agree on any pair of elements and 1 indicating that the two clusters are exactly the same. ARI is a corrected-for-chance version of the Rand Index, which equals 0 on average for random partitions, and 1 for two identical partitions. Therefore, the larger the ARI value is, the better the performance is. Mutual Information (MI) is a symmetric measure to quantify the statistical information shared between two distributions [10]. It can provide an indication of the shared information between a pair of clusterings. The second metric employed in this paper is the Normalized Mutual Information (NMI) introduced in [30], which is the MI between two clusterings normalized with the geometric mean of their entropies. NMI ranges from 0 to 1 and larger value indicates better performance.

Comparison Methods. Kohlschütter and Nejdl observed that the number of tokens in a text fragment, i.e. text density, is a valuable feature for segmentation decisions [19]. Therefore, they proposed a block fusion model that utilizes the text density ratios of subsequent blocks to identify segments, where the Web page segmentation problem is reduced to solving a 1D-partitioning task. Among the variants of their model, Bf-rulebased achieves the best performance. Bf-rulebased constrains the density-based fusion operation between subsequent blocks with a set of gap-enforcing tags and a set of gap-avoiding tags. We implemented Bf-rulebased for conducting comparison. The optimal fusing threshold ϑ_{max} is tuned with the training set of our data. Chakrabarti et al. proposed a graph-theoretic approach to deal with Web page segmentation [7]. They cast the problem as a minimization problem on a weighted graph with the nodes as the DOM tree nodes and the edge weights as the cost of placing the end nodes in the same segment or different segments. They presented a learning framework to learn these weights from manually labeled data. The proposed CCLus algorithm solves this problem with correlation clustering on a graph that only contains leaf DOM nodes of a page as the nodes of the weighted graph. The proposed GCUts algorithm solves this problem with energy-minimizing cuts on a graph that regards each DOM node as a node of the graph. GCUts involves a rendering constraint to ensure that, if the root node of a subtree is in a particular segment, all the nodes in the entire subtree are in the same segment. We implemented both CCLus and GCUts to conduct comparison. Our training data is employed to learn the feature weights as well as the trade-off parameter λ of two counterbalancing costs in the objective functions.

Table 2: Comparison of segmentation results on the entire data set.

	ARI	NMI
WPS_Slack	0.732	0.824
WPS_Margin	0.749	0.841
Bf-rulebased ($\vartheta_{max} = 0.65$)	0.605	0.761
CCLus ($\lambda = 0.62$)	0.489	0.662
GCUts ($\lambda = 0.53$)	0.630	0.770

7.2 Overall Segmentation Results

Recall that, in Section 6.1, we employ two quadratic forms of SVM model, namely, slack re-scaling and margin re-scaling, which incorporate informative segment oriented loss and Hamming loss respectively. Accordingly, we have two variants of our model, named WPS_Slack and WPS_Margin respectively. The learning precision ε for the feature weight estimation in Algorithm 2 is set to 0.1.

We first conduct experiment on the entire data set containing 1,000 pages. 4-fold cross-validation is employed and the average performance evaluated with ARI and NMI is reported in Table 2. Both variants of our model can outperform the comparison methods significantly. The improvements in ARI values over Bf-rulebased and GCUts are about 25%. In addition, paired t-tests (with $P < 0.01$) comparing the variants of our model with the comparison methods show that the performance of our variants is significantly better. Among different variants of our model, WPS_Margin with Hamming loss can achieve better performance than WPS_Slack with informative segment oriented loss. It is because WPS_Slack favors the segmentations that generate more accurate informative segments, which makes it perform less accurately on the uninformative segments. While the

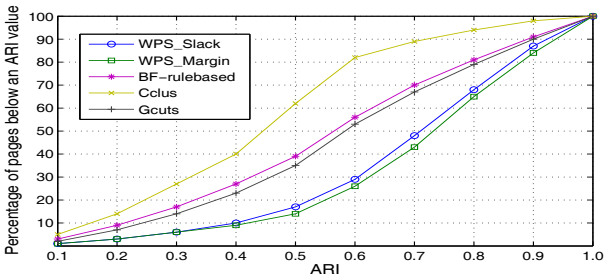


Figure 4: Percentage of pages below an ARI value.

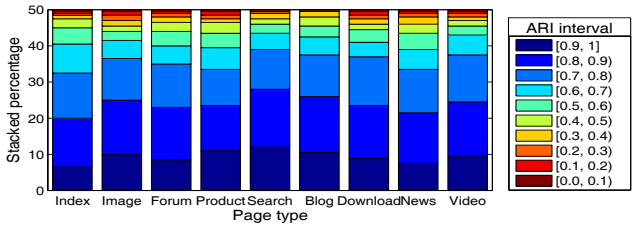


Figure 5: Stacked percentage in different ARI value intervals by WPS_Margin.

Table 3: Type-specific segmentation results in ARI.

Type ID	1	2	3	4	5	6	7	8	9
WPS_Slack	.745	.771	.761	.763	.802	.794	.769	.757	.782
WPS_Margin	.778	.795	.787	.786	.829	.819	.793	.775	.807
Bf-rulebased	.617	.640	.662	.657	.687	.671	.653	.701	.654
CClus	.528	.532	.529	.547	.564	.553	.540	.532	.543
GCuts	.674	.687	.665	.691	.703	.697	.684	.669	.698

Table 4: Results of informative block extraction.

Type ID	2	3	4	5	6	7	8	9	ALL\Index
WPS_Slack	.91	.90	.92	.93	.93	.90	.94	.90	.88
WPS_Margin	.88	.88	.86	.88	.88	.83	.85	.83	.84
Bf-rulebased	.62	.59	.63	.72	.75	.73	.75	.63	.68
GCuts	.64	.61	.60	.78	.78	.71	.71	.65	.70

evaluation metrics ARI and NMI do not consider the importance difference among the segments, which gives WPS_Margin more advantage in the reported performance.

GCuts achieved slightly better performance than Bf-rulebased. And CClus achieved the lowest accuracy, because it reports many non-rectangular segments since the built graph of it only contains the leaf DOM nodes. Generally, non-rectangular segment should not exist according to common sense. We observe that the page designers now prefer using and tags together with Cascading Style Sheets (CSS) in page design. This makes the heuristics based on gap-enforcing tags in Bf-rulebased less effective. Similar to our method, GCuts is more adaptable since its feature weights are tuned with training examples. It also solves the problem of reporting non-rectangular segments to some extent with the rendering constraint. The cumulative percentage of Web pages for which the segmentation performance of a particular method is less than a certain ARI value is plotted in Figure 4. The slower the curve goes up from left to right, the better the corresponding method is. For CClus, Bf-rulebased and GCuts, the percentages of ARI value lower than 0.6 are about 82%, 56% and 52% respectively. For the variants of our framework, such percentages are between 20% to 30%.

7.3 Type-specific Segmentation Results

To evaluate the type-specific performance of different segmentation methods, we employ the page set of each type as an individual experimental data set. Also 4-fold cross-validation is conducted on each of the nine page sets. The results evaluated with ARI are reported in Table 3. We find that all methods can achieve better performance on an individual page type compared with on the entire data set. This is because the trained or tuned parameters in different methods are more tailor-made for a particular type so as to achieve better accuracy. Among different types of pages, Search Result and Blog are relatively easier to handle. The main reason is that these two types of pages have relatively simple structures. Index and News are the most difficult types. The reason is that these two types of pages have more heterogenous structures and various information topics. In addition, the loss function in WPS_Slack is set to 1 for the index pages since no informative segments are annotated for them, which makes WPS_Slack less effective in tackling index pages compared with tackling the other types. Note that our framework does not have type-specific features, since we assume that the type of the pages is unknown. The stacked percentage in different ARI value intervals for individual page types is given in Figure 5. Besides the types of Product and News, the percentage of ARI value lower than 0.6 is no more than 20%.

7.4 Informative Segment Results

Recall that besides the index pages, we annotated informative segments in the pages of other types. We conduct evaluation on the performance of our method for segmenting these informative segments. If the informative segment of a page is accurately segmented, we regard this page successfully handled. If the informative segment is regarded as a subpart of any other segment or it is separated into several sub-segments, this page is not successfully handled. We calculate the percentage of the pages whose informative segments are successfully segmented.

In addition to the eight individual sets of pages, we have another data set that aggregates the pages of these eight types and it is called ALL\Index. The average results from 4-fold cross-validation are reported in Table 4. The WPS_Slack with informative segment oriented loss achieved the best performance and dominated other methods significantly. It demonstrates that the design of informative segment oriented loss is helpful for capturing informative segments for different types. It also shows that our variants with different loss designs have their own advantages to handle segmentation tasks with different focuses, making our framework more adaptable compared with previous works. The trained segmentation models on the individual types are more tailor-made so as to achieve better results compared with that on ALL\Index. After some manual checking, we found that most unsuccessful cases include some noise elements as part of the informative segments, such as the comments in the news and blog pages. In Bf-rulebased, one heuristic rule is that a segmentation gap should be enforced after the tags <h1>-<h6>. Consequently, Bf-rulebased always splits the title formatted with <h1>-<h6> of an informative segment and the main content of the segment, which results in over-segmentation mistakes.

Algorithm 3: In-segment position finding.

initialization: $s(t) \leftarrow \emptyset$, $s(b) \leftarrow \emptyset$, $s(l) \leftarrow \emptyset$, $s(r) \leftarrow \emptyset$, $s(m) \leftarrow \emptyset$, $\{b_i\}_{i=1}^n$: boundaries having the label Y in s
if $n = 0$ **then**
  $s(m) \leftarrow s$
else if $n = 1$ **then**
  if $\{b_i\}_{i=1}^n$ are horizontal **then**
    $s(t) \leftarrow \{s_1^o\}$, $s(b) \leftarrow \{s_1^b\}$
  else
    $s(l) \leftarrow \{s_1^o\}$, $s(r) \leftarrow \{s_1^b\}$
  end if
else
  if $n = 2$ **then**
    $b' \leftarrow b_1$, $b'' \leftarrow b_2$
  else
    calculate $F_i' = \langle \mathbf{\Psi}(b_i, \mathtt{Y}), \mathbf{w} \rangle$ for each b_i
    $b' \leftarrow \text{argmax}_{b_i} F_i'$, $b'' \leftarrow \text{argmax}_{b_j \in \{b_i\}_{i=1}^n \setminus b'} F_j'$
  end if
  if $\{b_i\}_{i=1}^n$ are horizontal **then**
    $s(t) \leftarrow$ the segments above b'
    $s(m) \leftarrow$ the segments between b' and b''
    $s(b) \leftarrow$ the segments below b''
  else
    $s(l) \leftarrow$ the segments on the left of b'
    $s(m) \leftarrow$ the segments between b' and b''
    $s(r) \leftarrow$ the segments on the right of b''
  end if
end if

8. APPLICATION IN WEB PAGE CLASSIFICATION

8.1 Feature Extraction for Classification

As discussed above, the output of our segmentation model provides a good structural analysis of Web pages enabling better utilization for different Web page mining tasks. Such efficacy of our model is examined in the task of page functional type classification [26]. Different from topical type, functional type describes the role that a Web page plays, such as image page mainly presenting an image, video page mainly presenting a video, etc. The identification of functional type is very useful for different Web mining problems. For example, search result ranking normally considers functional type as one factor. Page crawler can also trigger a better crawling strategy given the type of crawled pages.

The functional type of a page is closely related to the page structure and the functional terms appearing in different positions of the page. For example, the functional terms "reply" and "post" in the informative segment of a forum page are indicative features. The term "forum" in the header and bottom sections is also an important feature. To utilize the output of our segmentation model in this classification task, five different in-segment positions are defined, namely, top, bottom, left, right, and middle. For a segment s, these positions are denoted as $s(t)$, $s(b)$, $s(l)$, $s(r)$, and $s(m)$ respectively. Let $b_{1..n}$ denote the boundaries having the label Y inside s. The procedure of finding the in-segment positions of s is given in Algorithm 3. Note that an in-segment position can contain more than one subsegments when $n > 2$.

To construct the feature vector of each input page, we consider the in-segment positions in two major segments of the page obtained from different layers of the segmentation procedure. The first major segment is the entire page and the second major segment is the largest segment obtained by segmenting the entire page. In Figure 3, the major segments are s_0 in (a) and s_2^o in (b). Suppose b_1, b_2, b_3, and b_4 are labeled as Y by our segmentation model. Thus, s_0 is segmented into $s_0(t)$ containing s_1^o, $s_0(m)$ containing s_2^o, and $s_0(b)$ containing s_2^b as shown in Figure 3(b). s_2^o is segmented into $s_2^o(l)$ containing s_3^o, $s_2^o(m)$ containing s_4^o, and $s_2^o(r)$ containing s_4^b as shown in Figure 3(c). After a page is segmented by our model and the in-segment positions of the major segments are determined by Algorithm 3, we calculate the in-segment position based TF-ISF value for each term t in the individual positions, where TF is the frequency of t in this position of the page and ISF is the inverted segment frequency of t calculated based on the same position across the corpus. Therefore, a single term is decomposed into 10 different dimensions in the feature vector according to its in-segment positions in the two major segments. After some basic preprocessing such as stop word removal, we perform feature selection with information gain (IG) [36]. Only the top 10% of terms are retained in the construction of the feature vector so as to control the dimensionality.

Table 5: The results of page classification.

	Precision	Recall	F1
Our method	0.945	0.926	0.936
Unstructured_Text	0.698	0.673	0.685
Bf-rulebased_Segment	0.755	0.735	0.745
GCuts_Segment	0.772	0.745	0.758

8.2 Experimental Setting and Results

We prepare another collection of 4,000 pages from the eight types having informative segments as indicated in Table 4. Each type contains about 500 pages. LibSVM [8] with linear kernel is employed to train eight classifiers under one-against-the-rest strategy for multi-class classification. To conduct comparison, three baseline methods are designed. The first baseline, named Unstructured_Text, is a purely text-based method without considering page structure. It employs all terms after preprocessing to form the feature vectors. We design the other two comparison methods based on the informative segments detected by Bf-rulebased and GCuts. The informative segment of each page is identified with the following rule. The largest segment that appears (maybe partially) in the first screen of a page is regarded as its informative segment. The first screen of a page is defined as the top fraction of the page with the height of 1,000 pixels. Then, the feature vector of a page employs the terms appearing in its informative segment as the dimensions. These two baselines are named Bf-rulebased_Segment and GCuts_Segment respectively. For our method, the employed segmentation model is the variant of WPS with margin re-scaling. All the segmentation models are tuned or trained with the entire data set in Table 1.

The average results, evaluated with macro-averaged Precision, Recall and F1 measure, of 4-fold cross-validation are reported in Table 5. Our method achieves significantly better performance compared with the other methods. The percentages of improvements in F1 are about 23% to 37%. This demonstrates that the page structure information, revealed by the in-segment positions in our method, is very useful in the classification of page functional types. The F1 values of different page types are given in Figure 6. We observe that Bf-rulebased_Segment and GCuts_Segment face more difficulties in handling three types of pages, namely, Blog,

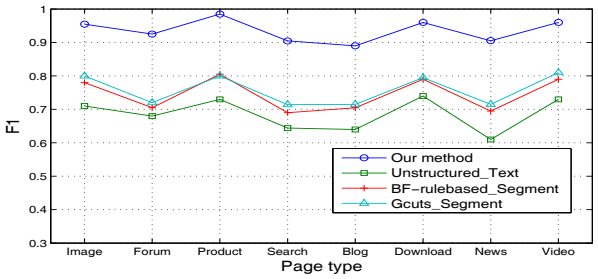


Figure 6: Classification performance for different page types.

News, and Search Result. It is because these types of pages have very few functional terms in their main content. The two informative-segment-based methods cannot well distinguish the main content of a news page and that of a blog page. Although the baseline Unstructured_Text keeps the functional terms outside the informative segments, it does not have the structure information to differentiate the appearances of the same term as functional term in a specific position and as a normal term in the main content. Therefore, its performance is even degraded by the negative effect of the noise.

9. CONCLUSIONS

We propose a framework which can perform page segmentation with a structured prediction approach. The segmentation task is formulated as a structured labeling problem on the WPS-graph. Each labeling scheme on the WPS-graph corresponds to a possible segmentation of the page. The feature weight learning algorithm is developed based on the structured output Support Vector Machine framework so that it is able to consider the inter-dependency among the vertices of a WPS-graph. Extensive experiments demonstrate that our framework achieves better performance compared with state-of-the-art methods.

10. REFERENCES

[1] S. Baluja. Browsing on small screens: recasting web-page segmentation into an efficient machine learning framework. In *WWW*, pages 33–42, 2006.

[2] Z. Bar-Yossef and S. Rajagopalan. Template detection via data mining and its applications. In *WWW*, pages 580–591, 2002.

[3] L. Bing, W. Lam, and Y. Gu. Towards a unified solution: Data record region detection and segmentation. In *CIKM*, pages 1265–1274, 2011.

[4] L. Bing, W. Lam, and T.-L. Wong. Robust detection of semi-structured web records using a dom structure-knowledge-driven model. *ACM Trans. Web*, 7(4):21:1–21:32, 2013.

[5] L. Bing, W. Lam, and T.-L. Wong. Wikipedia entity expansion and attribute extraction from the web using semi-supervised learning. In *WSDM*, pages 567–576, 2013.

[6] D. Cai, S. Yu, J.-R. Wen, and W.-Y. Ma. VIPS: a vision-based page segmentation algorithm. Technical report, Microsoft (MSR-TR-2003-79), 2003.

[7] D. Chakrabarti, R. Kumar, and K. Punera. A graph-theoretic approach to webpage segmentation. In *WWW*, pages 377–386, 2008.

[8] C.-C. Chang and C.-J. Lin. Libsvm: A library for support vector machines. *ACM Trans. Intell. Syst. Technol.*, 2(3):27:1–27:27, 2011.

[9] Y. Chen, W.-Y. Ma, and H.-J. Zhang. Detecting web page structure for adaptive viewing on small form factor devices. In *WWW*, pages 225–233, 2003.

[10] T. M. Cover and J. A. Thomas. *Elements of information theory*. 1991.

[11] G. B. Dantzig and M. N. Thapa. *Linear Programming 1: Introduction*. Springer-Verlag New York, Inc., 1997.

[12] E. S. de Moura, D. Fernandes, B. Ribeiro-Neto, A. S. da Silva, and M. A. Gonçalves. Using structural information to improve search in web collections. *J. Am. Soc. Inf. Sci. Technol.*, 61(12):2503–2513, 2010.

[13] S. Debnath, P. Mitra, N. Pal, and C. L. Giles. Automatic identification of informative sections of web pages. *IEEE Trans. on Knowl. and Data Eng.*, 17(9):1233–1246, 2005.

[14] D. Fernandes, E. S. de Moura, A. S. da Silva, B. Ribeiro-Neto, and E. Braga. A site oriented method for segmenting web pages. In *SIGIR*, pages 215–224, 2011.

[15] S. D. Gollapalli, C. Caragea, P. Mitra, and C. L. Giles. Researcher homepage classification using unlabeled data. In *WWW*, pages 471–482, 2013.

[16] Q. Hao, R. Cai, Y. Pang, and L. Zhang. From one tree to a forest: a unified solution for structured web data extraction. In *SIGIR*, pages 775–784, 2011.

[17] G. Hattori, K. Hoashi, K. Matsumoto, and F. Sugaya. Robust web page segmentation for mobile terminal using content-distances and page layout information. In *WWW*, pages 361–370, 2007.

[18] L. Hubert and P. Arabie. Comparing partitions. *Journal of Classification*, 2(1):193–218, 1985.

[19] C. Kohlschütter and W. Nejdl. A densitometric approach to web page segmentation. In *CIKM*, pages 1173–1182, 2008.

[20] H. Lam and P. Baudisch. Summary thumbnails: readable overviews for small screen web browsers. In *CHI*, pages 681–690, 2005.

[21] J. Lazar, A. Allen, J. Kleinman, and C. Malarkey. What frustrates screen reader users on the web: A study of 100 blind users. *Int. J. Hum. Comput. Interaction*, 22(3):247–269, 2007.

[22] X. Li, T.-H. Phang, M. Hu, and B. Liu. Using micro information units for internet search. In *CIKM*, pages 566–573, 2002.

[23] S.-H. Lin and J.-M. Ho. Discovering informative content blocks from web documents. In *KDD*, pages 588–593, 2002.

[24] C. Lu, L. Bing, and W. Lam. Structured positional entity language model for enterprise entity retrieval. In *CIKM*, pages 129–138, 2013.

[25] J. Pasternack and D. Roth. Extracting article text from the web with maximum subsequence segmentation. In *WWW*, pages 971–980, 2009.

[26] X. Qi and B. D. Davison. Web page classification: Features and algorithms. *ACM Comput. Surv.*, 41(2):12:1–12:31, 2009.

[27] K. Radinsky and P. N. Bennett. Predicting content change on the web. In *WSDM*, pages 415–424, 2013.

[28] W. M. Rand. Objective Criteria for the Evaluation of Clustering Methods. *Journal of the American Statistical Association*, 66(336):846–850, 1971.

[29] R. Song, H. Liu, J.-R. Wen, and W.-Y. Ma. Learning block importance models for web pages. In *WWW*, pages 203–211, 2004.

[30] A. Strehl and J. Ghosh. Cluster ensembles — a knowledge reuse framework for combining multiple partitions. *J. Mach. Learn. Res.*, 3:583–617, 2003.

[31] F. Sun, D. Song, and L. Liao. Dom based content extraction via text density. In *SIGIR*, pages 245–254, 2011.

[32] I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, 6:1453–1484, 2005.

[33] K. Vieira, A. S. da Silva, N. Pinto, E. S. de Moura, J. a. M. B. Cavalcanti, and J. Freire. A fast and robust method for web page template detection and removal. In *CIKM*, pages 258–267, 2006.

[34] J. Wang, C. Chen, C. Wang, J. Pei, J. Bu, Z. Guan, and W. V. Zhang. Can we learn a template-independent wrapper for news article extraction from a single training site? In *KDD*, pages 1345–1354, 2009.

[35] L. A. Wolsey. *Integer programming*. Wiley, 1998.

[36] Y. Yang and J. O. Pedersen. A comparative study on feature selection in text categorization. In *ICML*, pages 412–420, 1997.

[37] L. Yi, B. Liu, and X. Li. Eliminating noisy information in web pages for data mining. In *KDD*, pages 296–305, 2003.