# Research of Adaptive Transformation Method Based on Webpage Semantic Features for Small-Screen Terminals

Hao Li, Qingtang Liu, Min Hu, and Xiaoliang Zhu

**Small-screen mobile terminals have difficulty accessing existing Web resources designed for large-screen devices. This paper presents an adaptive transformation method based on webpage semantic features to solve this problem. According to the text density and link density features of the webpages, the webpages are divided into two types: index and content. Our method uses an index-based webpage transformation algorithm and a content-based webpage transformation algorithm. Experiment results demonstrate that our adaptive transformation method is not dependent on specific software and webpage templates, and it is capable of enhancing Web content adaptation on small-screen terminals.**

**Keywords: Webpage transformation, mobile terminals, semantic features, text density, link density.**

## I. Introduction

As a result of the proliferation of wireless handheld devices, mobile terminals have become indispensable tools in daily life. However, in the era of rapid development of mobile phones, when mobile devices attempt to access mass Internet resources, which were originally designed for large-screen computers, it is clear that mobile terminal screens are too small, the Internet connection speed is slow, the storage space is limited, the computational capability is weak, the battery standby time is short, and so on [1].

To adaptively rendering webpage content on small-screen devices, Carnegie Mellon University (CMU), Pittsburgh, PA, USA, proposed a thumbnail method to enhance mobile Internet access quality of service [2]. This method preserves the original content of webpages well, allowing users to accurately grasp the entire webpage structure; when users browse familiar webpages, this method is especially effective. However, when a page is too large, the converted page becomes too small, so users have difficulty clearly browsing the specific contents. Most Internet resources are currently in the form of webpages, which were written in the HTML language. Special standards were set for mobile Internet, according to the features of handheld devices. These standards were either the HTML standard reduction, such as CHTML (Compact HyperText Markup Language) and HDML (Handheld Devices Markup Language) [3], or completely new standards, such as WML (Wireless Markup Language Wireless Markup Language) [4]. According to these new standards, rewriting the existing webpages is simple and easy. For the majority of the existing

Manuscript received Dec. 7, 2012; revised Mar. 1, 2013; accepted Mar. 21, 2013.

This work was supported by the Chinese Key Projects in the National Science & Technology Pillar Program (NO.2012BAD35B02).

Hao Li (phone: +86 15072411065, lihao.0205@gmail.com), Min Hu (sophie423@163.com), and Xiaoliang Zhu (zhuxl@mail.ccnu.edu.cn) are with the National Engineering Research Center for E-Learning, Huazhong Normal University, Wuhan, China.

Qingtang Liu (liuqtang@mail.ccnu.edu.cn) is with the Colleage of Information and Journalism Communication, Huazhong Normal University, Wuhan, China.

http://dx.doi.org/10.4218/etrij.13.0112.0834

Internet webpage resources, such a rewriting method would undoubtedly be cumbersome and labor-intensive.

To provide webpage content in an intelligent layout for small-screen terminals, in 2003, Microsoft Asia Research Institute presented the VIsion-based Page Segmentation (VIPS) algorithm, used to extract the semantic structure of a given webpage [5]. The VIPS algorithm represented such methods based on the page visualization information for segmentation; it extracted the structure and visual information from the webpages based on heuristic rules and then divided webpages into small blocks. This method is simple, easy to implement, and has advanced efficiency, but it does not have general applicability. Baluja transformed webpage segmentation into an efficient machine-learning framework that segmented the pages into blocks by a decision tree and the decision tree information entropy reduction [6]. The method converted the page segmentation process into a decision tree classification process. Although this method has a strong mathematical theoretical basis because it involves correlation judgment, it cannot guarantee the correct rate of the page segmentation. In 2008, Chakrabarti and others presented a webpage segmentation method based on graph theory, and they were the first to transform the page segmentation problem into a weighted graph combinatorial optimization problem [7]. This method can be applied to all pages of the Web, so it has general applicability. However, as a result of the graph construction, the graph on behalf of a webpage is very large, so the graph partitioning process has lower efficiency and is impractical. Xiang and Shi predefined some universal patterns and then searched for these patterns in the layout trees [8]. After conversion, pages could basically keep the visual effects of the original pages, but the tasks of web classification and pattern matching were dramatically heavy.

To solve the problem that small-screen mobile terminals have difficulty accessing existing web resources, this paper presents an adaptive transformation method based on webpage semantic features (WSFs). According to the text density and link density features, webpages are divided into index webpages and content webpages. We propose a content-based webpage transformation algorithm and an index-based webpage transformation algorithm. The structure of the paper is as follows. The first section introduces the aims of the method and the related works. The second section describes the design of the webpage adaptive transformation method based on WSFs. The third section presents our designed experiment and results, and the paper ends with a summary and a conclusion.

## II. Methodology

As shown in Fig. 1, the processing flow of our method includes the following steps. 1) Data preprocessing: this step mainly includes the URL format examination, HTML structured regularization, and HTML tag format tidiness and Document Object Model (DOM) tree representation of the webpage document. 2) Semantic structural features extraction of webpage: this process includes the calculation of text density and link density of the page, to represent the tag level and structure level feature of the webpage. 3) Judgment of webpage type: the pages are divided into index type and content type in our method. 4) Webpage adaptive transformation: to deal with the two different types of webpage, we propose a content-based webpage transformation algorithm and an index-based webpage transformation algorithm. 5) Webpage rendering: after completion of the webpage transformation, this phase includes web integration and rendering for the users.

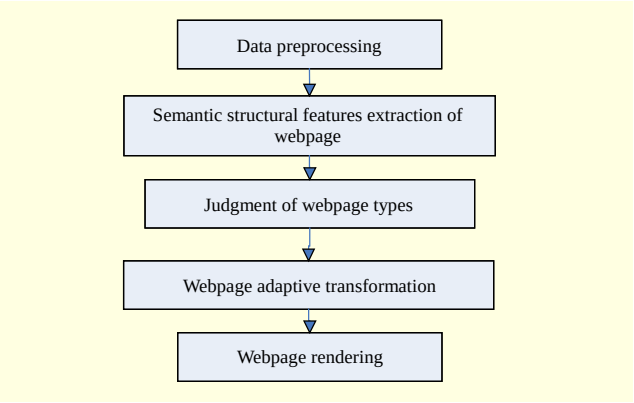

Fig. 1. Processing flow chart.

### 1. Representing Webpage Features

For a webpage, we establish a web semantic model, as shown in Fig. 2, which has six semantic levels from the bottom to the top [9]. 1) Text level: the source code of the webpage, for example, <li class="leaf first"> <a href="/about-sakai" title="About Sakai"> About </ a> </ li>. 2) Tag level: HTML elements form the building blocks of all websites; for example, <table> represents a form, <tr> denotes a row in a table. 3) HTML structure level: HTML tags also have layout characteristics, which means that some HTML tags could very well reflect the page layout and structure information, such as <table> and  labels being used for webpage layout. 4) HTML rendering structure: Webpages will eventually be presented to the users graphically, and all the HTML tags are rendered by the browser; So, this graphical interface reflects the rendering structure of the HTML webpages; For example, for a specific page, the top region is the navigation bar, the central area is the body content, and the bottom zone is the copyright and contact information. 5) Writing structure level: When page

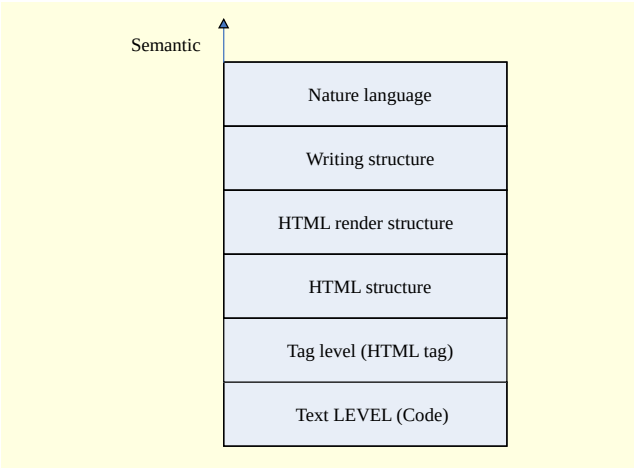


Fig. 2. Webpage semantic model figure.

encoders write web functions, their personal writing habits are reflected in the webpage; The characteristics of writing habits constitute the web writing structural features; For example, there may be a text description at the bottom of a picture. 6) Natural language level: Web content may expose some semantic intentions of the page developers; For example, for the main body of a webpage, a subblock area may be notification information, and another subblock zone may be recent news information. In this paper, we will focus on the first three semantic levels.

At the webpage text level, as a result of HTML non-normative writing, we use Jtidy [10] to clean up its format, process non-closed labels, and so on. At the webpage tag level, we mainly focus on the content of text tags of webpages, so we use text density $\delta(b)$ to ration the characteristics of the tags of the HTML pages [11]. The text density of current node, $\delta(b)$, is given as

$$\delta(b)=\begin{cases}\dfrac{T'(b)}{(L(b)-1)*\text{maxLen}}, & L(b)>1,\\ \dfrac{T(b)}{\text{maxLen}}, & L(b)\le 1,\end{cases} \quad (1)$$

where $T(b)$ denotes the length of the text of the current node, $L(b)$ represents the number of text rows of the current node, $T'(b)$ denotes the length of the text of the current node when the number of text rows is larger than 1, and maxLen represents the one-line text length of the current screen size. Regarding the structure of the page, we mainly focus on link density. The link density of the current node, $\theta(b)$, is given as

$$\theta(b)=\frac{T_a(b)}{T(b)}, \quad (2)$$

where $T_a(b)$ denotes the text length sum of all <a> tags of the current node and its children nodes.

## 2. Algorithm Flow

In the preprocessing stage, URL address validation is performed first, and then the webpage character encoder is detected so that the transformed webpages are not garbled. In the next step, remote URL HTML content is copied and stored locally. In Step 4, the HTML tidy interface is recalled to clean the HTML tags [10]. In the phase of converting webpages into an HTML DOM tree, we recall the HTML Agility Pack interface to implement its function [12]. An HTMLInput object based on the input URL is constructed. Then, the parse method is called to generate an HtmlDocument object, which contains all the attributes of the current webpage. Based on the HtmlDocument object, we propose an algorithm to determine what type of webpage the current page is. A webpage is either an index webpage or a content webpage. An index-based webpage transformation algorithm and a content-based webpage transformation algorithm are used to extract the main content of the current webpage. In the page rendering stage, adaptive webpage content is presented to the users according to the screen size and other preferences of the terminal devices. The specific pseudocode is as follows.

**Algorithm 1.** Adaptive transformation algorithm based on webpage sematic features.
**Input**: URL
**Output**: adaptively transformed content webpage for the small-screen terminals
**Algorithm**:

```
//Step 1: URL format standardization
   public string urlformat(string url){
   return urlformat (url);}

//Step 2: Character encoding detection
   public string Characterdetect(string url){
   return mozillaCharDetect(url);}

//Step 3: Get HTML source code
   public Stream getHtmlCode(string url; string character){
   return GetRemoteStream(url, character);}

//Step 4: HTML tag tidy
   public string htmltidy(Stream remoteStream){
   return htmlTidy(remoteStream);}

//Step 5: Transformation HTML page into DOM tree
   public HtmlDocument htmltidy(string htmlsource){
   HTMLInput Input = new URLInput(htmlsource);
   HtmlDocument document = Input.Parse();
   Return document;}

//Step 6: Judgment of webpage type.
   public string judgeOfWebType(HtmlDocument document){
```

```
    return judgewebtype(document);}

//Step 7: Call the webpage transformation algorithm
    public string webTransform(HtmlDocument document, string
    type){
        If(type==“index”)
        Then return indexWebTransform(document);
        If(type==“content”)
        Then return contentWebTransform(document);}

//Step 8: adaptive webpage rendering for the small-screen terminals
    public string adaptedWebContent (string webcontent,parameters
    terminal){
    return adaptedWebContent(webcontent, terminal)}
```

## 3. Algorithm for Judgment of Webpage Type

Webpages are determined to be either content webpages or index webpages according to their presented form. On an index webpage, content is displayed in subregions, each of which has a custom abstract theme. Each zone of a webpage has multiple rows, each row of content having an identified specific topic and each topic linked with another webpage, as shown in Fig. 3. On a content webpage, a specific topic is described by an apparent complete title, and there is a large section of text to describe this specific topic in detail or there may be additional topics linked to this topic, as shown in Fig. 4. We use webpage text density and webpage link density to judge the webpage type. The webpage text density ($D_t$) calculation method and the webpage link density ($D_l$) calculation method are respectively shown as follows:

$$D_t = L_n / N, \tag{3}$$

$$D_l = L_l * C_l / N. \tag{4}$$

In (3), $L_n$ denotes the sum of the length of all no-link text for the current web, and $N$ represents the total number of nodes of the current webpage. In (4), $L_l$ represents the sum of text length of all <a> tags for the current webpage, and $C_l$ denotes the total link number of the current webpage. Based on a statistical analysis method, we mainly use the ratio of the whole webpage text density and the link density to judge the webpage type, as

$$D_l / D_t = L_l * C_l / L_n, \tag{5}$$

where $D_l/D_t$ represents the link density and text density ratio of this page.

We use the text density and link density of the webpage nodes to extract the main content for the content-based webpage, which will be discussed in detail in subsection II.5, and the related calculation methods are shown in (1) and (2), respectively.

We randomly select 150 content webpages and 150 index webpages and calculate the text density and link density, respectively, as shown in Fig. 5. From the figure, it is clear that there is a polarization for the page link density with text density ratio distribution. When the link density with a text density ratio of the current webpage is greater than the threshold value, the webpage is an index webpage; otherwise, it is a content webpage (the threshold value is represented by the red line in


Fig. 3. Index webpage. (Screenshot from www.chinadaily.com.cn.)


Fig. 4. Content webpage. (Screenshot from www.chinadaily.com.cn/china/2012cpc/2012-11/04/content_15872387.htm.)

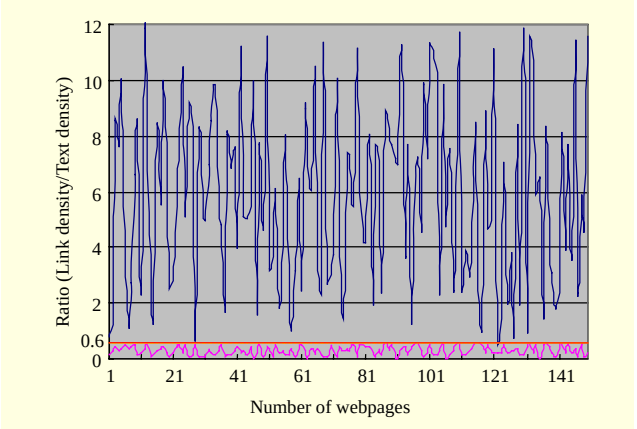


Fig. 5. Webpage link density with text density ratio distribution.

Fig. 5). The specific pseudocode of this algorithm is as follows.

```
Algorithm 2. Algorithm for judgment of webpage type.
Input: htmlDocument
Output: webpage type;
Algorithm:
//Step 1: get the node list of the web document
  Public IList<HtmlNode> getNodeList (HtmlDocument document){
  HtmlNode root= document.DocumentNode;
  IList<HtmlNode> allnodes = root.SelectNodes("//*");
  Return allnodes;}

//Step 2: get the length of no link text
  Public int getNoLinkText(IList<HtmlNode> allnodes){
    Int htmlNodeCount= allnodes.count; Int nolinkLen=0;
      For(int i=0; i< htmlNodeCount; i++){
      Htmlnode current = allnodes [i];
        If(current.child.count==0) &&( current.type=="Text")
        Then nolinkLen = nolinkLen+ current.InnerText.Length;
      If(current.child.count >0)
      Then scanNolink(node, nolinkLen); }
      // Traversing iterate the current node
      return nolinkLen;}

//Step 3: get the A tag count of the web document
  Public int getAtagCount(htmlNode root){
    IList<HtmlNode> aTagnodes = root.SelectNodes(".//a[@href]");
    Return aTagnodes.count;}

//Step 4: get the length of link word
  Public int getAtagTextLen(IList<HtmlNode> aTagnodes){
    Int aTagCount= aTagnodes.count;Int linkLen=0;
      For(int i=0; i< aTagCount; i++){
      Htmlnode current = aTagnodes [i];
      If(aTagnodes [i].InnerText.Length>0)
      Then int linkLen = linkLen + aTagnodes [i].InnerText. Length;
      return linkLen;}

//Step 5: count the link density/text density
  Public Double denstiyRation(){
    htmlNodeCount =getNodeList(root).count;
  //5.1count link density
    linkCount = getAtagCount(aTagnodes);
    LinkWordLen= getAtagTextLen(aTagnodes)
    Double linkdensity= linkCount* LinkWordLen / htmlNodeCount;
  //5.2count no link text density
    nolinkLen = getNoLinkText (allnodes);
    Double textdensity= nolinkLen / htmlNodeCount
  //5.3 count the link density/text density
  Return (linkdensity/textdensity);}

//Step 6 judge the webpage type
  String judge(double ration){
    If(linkdensity /textdensity >thtreshold)   Then return "index";
    Else return "content";}
```

## 4. Index-Based Webpage Transformation Algorithm

For index webpages, we first use the VIPS algorithm [13] to segment the current webpage into small blocks and receive a block list. In 2003, the Microsoft Asia Research Institute proposed the VIPS algorithm to extract the semantic structure for a webpage [5]. The VIPS algorithm makes full use of page layout feature separators, and it employs a top-down approach, which is very effective. Compared to other webpage segmentation algorithms, VIPS is more stable and easier to apply to our research, so we utilize it in our study. According to the block list, a block type is set to each block; the block types include advertise, noise, navigator, and main content. Another page is selected with the same domain and website hierarchy to detect the navigator. The same sub-DOM trees are extracted as the navigator candidates from the selected two pages. Then, nodes without links and nodes that have links whose URL domain is outside of the domain of the current webpage are removed, so the remaining DOM nodes are set as the navigator type [13]. If the current block type is noise or advertise, the block is filtered out directly. Otherwise, we detect each block to search for a subtitle; if one exists, it will be extracted. The subtitle extraction is used to enable users to quickly find the content that matches their interest after the adapted transformation. If the current block type is navigator, HTML code is generated according to the text and URL of the current block. If the block type is main content, text and URLs of blocks are extracted from web content. After the mergence of the extracted text, the URL, and the block subtitle, the adaptive webpages are displayed on the small-screen terminals. The pseudocodes are as follows.

```
Algorithm 3. Index-based webpage transformation algorithm.
Input: HtmlDocument
Output: adaptively transformaed index webpage
Algorithm:
//Step 1: call the webpage fragmentation algorithm to segment the
          webpage
  Public List<block> pageSegmentation(HtmlDocument document){
    Return VIPS(document);}

//Step 2: noise block filter
  Public List<block> blocksFilter(List<block> segmentation){
    For(int i=0; i<segmentation.length; i++)
  {block current=segmentation [i];
    If (current.type=="advertise")   Then segmentation.remove(i);
    If (current.type=="noise")   Then segmentation.remove(i);}
    Return segmentation;}
```

```
//Step 3: call the subtitle extraction algorithm
  Public list<string> subtitleDetect(List<block> segmentation)
    {return subtitleDetect(segmentation);}

//Step 4: display of adapted content
  Public string adaptiveTrams(List<block> segmentation;
  list<string> sub){
    Stritn htmlcode; Int subTitleIndex=0;
    For(int i=0; i<segmentation.size; i++)
    {block current=segmentation [i];
  //4.1: webpage navigator content generation
    If(current.type==“navigator”) Then
    {List<rescord> resc=current.getRecord
    for(int k=0; k< resc.length; k++)
    htmlcode = htmlcode +Htmlgeneration(url,text);}
  //4.2: webpage main content generation
    If(current.type==“maincontent”) Then
    {List<rescord> resc=current.getRecord
    for(int k=0; k< resc.length; k++)
    htmlcode = htmlcode +Htmlgeneration(sub[subTitleIndex],
    url,text);
    subTitleIndex++}}
    Return htmlcode; }
```

## 5. Content-Based Webpage Transformation Algorithm

For content webpages, we mainly extract the body content of the page and then adaptively display it on small-screen terminals. We first obtain all HtmlNodes of the current webpage. Afterward, we traverse all nodes of the DOM tree of the current webpage and calculate the text density and link density of each node. If the text density of the current node is greater than the minimum text density threshold value and the link density is less than the minimum link density threshold, the current node is a main content node; otherwise, the current node is a noise node or not related to page topic content, which is directly filtered out in this paper. After the traversal of the whole DOM tree of the current page, all nodes related to the body content are extracted. Finally, upon completing the text and URL extraction and the page reorganization, the adapted webpages are displayed on the small-screen terminals. The pseudocodes are as follows.

**Algorithm 4.** Content-based webpage transformation algorithm.
**Input:** htmlDocument
**Output:** adaptively transformed content webpage
**Algorithm:**

```
//Step 1: get the node list of the web document
  Public IList<HtmlNode> getNodeList (HtmlDocument
  document){
  HtmlNode root= document.DocumentNode;
    IList<HtmlNode> allnodes = root.SelectNodes(“//*”);
      Return allnodes;}

// Step 2: main content extraction
  Public List<node> maincontent(IList<HtmlNode> nodes){
  List<node> maincont;
    For(int i=0;i<nodes.length;i++)
    {HtmlNode current= nodes [i];
      If (current.linkdensity<=min_link_density)&&
  (current.contentdensity>min_ content_density)
    Then maincont.add(current);}
      Return maincont;}

//Step 3: display of adapted content
  Public string adaptiveTrams(List < node > nodes, List<block>
  navigator){
    Stritn htmlcode;
    For(int i=0;i< navigator.length;i++)
      {htmlcode = htmlcode +Htmlgeneration(navigator [i])}
    For(int k=0; k< nodes.length; k++)
      {htmlcode = htmlcode +Htmlgeneration(nodes [i])}
    Return htmlcode;}
```

# III. Experiment Discussion

## 1. System Implementation

We use a proxy-based approach to implement our system, as shown in Fig. 6. We design a series of experiments to verify the effectiveness of our proposed method. We consider the judgment of webpage type, the generation of index webpages, the generation of content webpages, the effectiveness of the system validation of our proposed algorithm, and a comparison to other methods. Our adaptive services are deployed on an Apache server, and the proxy server is configured as follows: the CPU is an Intel Pentium (R) Dual E2180 2.00 GHz, the memory is PDDR2 SDRAM 667 MHz 1.0 GB, the hard disk is 150 GB with Seagate ST3160815AS, and the operating system is Microsoft Windows server 2003.

## 2. Judgment of Webpage Type

We randomly select 1,402 pages to prove the accuracy of the algorithm for judging webpage type. For those 1,402 test webpages, we first manually judge the page type and create test data sets. We design a series of experiments to verify the effectiveness of our proposed method. Then, we use an

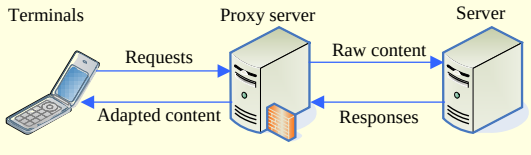


Fig. 6. System architecture.

Table 1. Accuracy of judgment of webpate type.

| Web Catalog | Website | Webpage count | Correctly judged count | Correctly judged ration | Webpage type |
|---|---|---|---|---|---|
| Academia | Harvard | 15 | 14 | 93.33% | index |
| | | 52 | 50 | 96.15% | content |
| | MIT | 18 | 17 | 94.44% | index |
| | | 62 | 61 | 98.39% | content |
| | Cambridge | 19 | 18 | 94.74% | index |
| | | 43 | 41 | 95.35% | content |
| Business | Microsoft | 21 | 20 | 95.24% | index |
| | | 45 | 43 | 95.56% | content |
| | IBM | 23 | 21 | 91.30% | index |
| | | 43 | 41 | 95.35% | content |
| | Lenovo | 17 | 15 | 88.24% | index |
| | | 44 | 42 | 95.45% | content |
| | business.sohu | 24 | 22 | 91.67% | index |
| | | 45 | 41 | 91.11% | content |
| | canon | 22 | 20 | 90.91% | index |
| | | 45 | 40 | 88.89% | content |
| Finance | eastmoney | 12 | 10 | 83.33% | index |
| | | 43 | 39 | 90.70% | content |
| | wsj | 12 | 11 | 91.67% | index |
| | | 45 | 41 | 91.11% | content |
| | finance.sina | 11 | 10 | 90.91% | index |
| | | 43 | 39 | 90.70% | content |
| | stockstar | 14 | 11 | 78.57% | index |
| | | 56 | 51 | 91.07% | content |
| | money.163 | 12 | 10 | 83.33% | index |
| | | 34 | 31 | 91.18% | content |
| Entertainment | 67.com | 11 | 9 | 81.82% | index |
| | | 43 | 41 | 95.35% | content |
| | ent.sina | 11 | 9 | 81.82% | index |
| | | 15 | 11 | 73.33% | content |
| | ent.huanqiu | 12 | 10 | 83.33% | index |
| | | 23 | 21 | 91.30% | content |
| | ent.sohu | 14 | 12 | 85.71% | index |
| | | 23 | 21 | 91.30% | content |
| News | news.sina | 14 | 12 | 85.71% | index |
| | | 25 | 23 | 92.00% | content |
| | news.sohu | 16 | 14 | 87.50% | index |
| | | 35 | 32 | 91.43% | content |
| | ifeng | 11 | 10 | 90.91% | index |
| | | 35 | 32 | 91.43% | content |


Fig. 7. Contrast effect diagrams before and after the adaptive transformation of index webpage. (Screenshots from www.sina.com.cn.)

algorithm to judge the page type to determine the page type automatically. The 1,402 test webpages are divided into the following portal categories: academia, business, finance, entertainment, and news. Table 1 shows the accuracy of the judgment of webpage type for each category.

## 3. Generation of Index Webpages

There are vast amounts of links and abstract text in the index webpages, which are mixed with an abundance of advertisement and other noise content. On the one hand, this noise content does not reflect the user's interest; on the other hand, this noise content affects the way the user accesses the information on the page. Therefore, during the adaptive phase, we filter out all the noise content. Therefore, users can quickly access the content they are interested in, and network traffic is significantly reduced. Taking the Sina website for example, as shown in Fig. 7, the left image shows the effect of the original pages in a personal computer with a large screen, the upper right image is the result with a normal IE browser using our adaptive transformation method, and the image in the lower right displays the result on the HTC A3366 mobile smartphone using our adaptive conversion method.

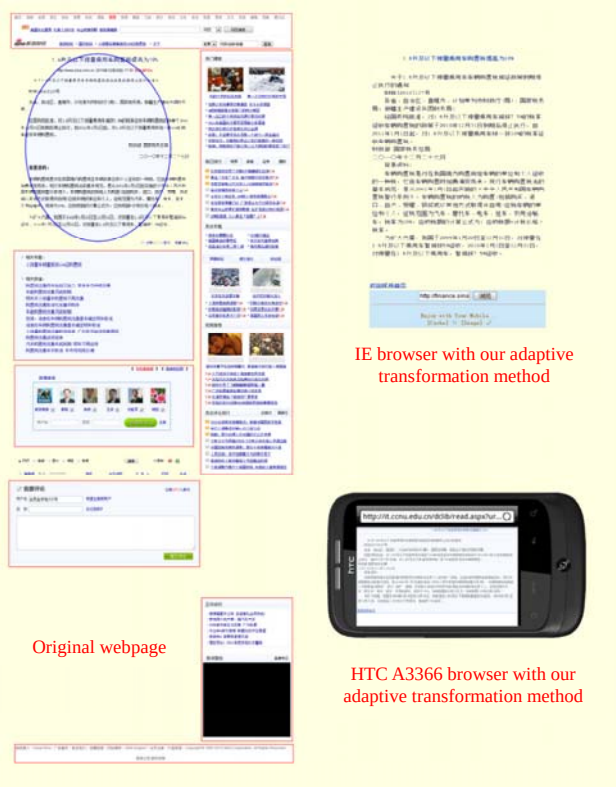


Fig. 8. Contrast effect diagrams before and after adaptive transformation of content webpage. (Screenshots from http://finance.sina.com.cn/g/20101228/17359175742.shtml.)

## 4. Generation of Content Webpages

For a content-based webpage, the user's actual interests correlate to the main content block of the webpage; therefore, blocks irrelevant to the main content do not draw the user's attention. The content enclosed by a blue circle in Fig. 8 is the main focus of the user. The user has minimal interest in the material set off by the red boxes in Fig. 8, so such material is directly filtered out in the adapted display processing phase of our method. A comparison of the webpages before and after adaptive transformation is shown in Fig. 8. The left image shows the effect of the original pages in a personal computer with a large screen, the upper right image shows the effect with a common IE browser using our adapted transformation method, and the lower right image shows the effect on an HTC A3366 mobile smartphone using our adaptive transformation method.

## 5. Qualitative Evaluation

For each webpage in our test set, we retrieve it on both a desktop browser and a mobile screen, using our content adaptation service. Displays on both screens are saved as a pair to use as a test case. We conduct an experiment with 30 volunteers, who verify the entire package of 200 test cases. The volunteers are juniors at Central China Normal University, all of whom have the habit of accessing the Internet via mobile devices and PCs. We provide all volunteers with an HTC A3366 mobile smartphone with a screen size of 3.2 inches. A tutorial is given to the students prior to the experiment. For each test case, a student volunteer is asked to compare the original desktop screen shot and the adapted mobile smartphone screen shot and rate the adaptation performance according to the following levels: satisfactory, tolerable, and unacceptable. For each level, the volunteer is asked to justify his/her rating in detail. In this way, we ensure that their rating is reliable.

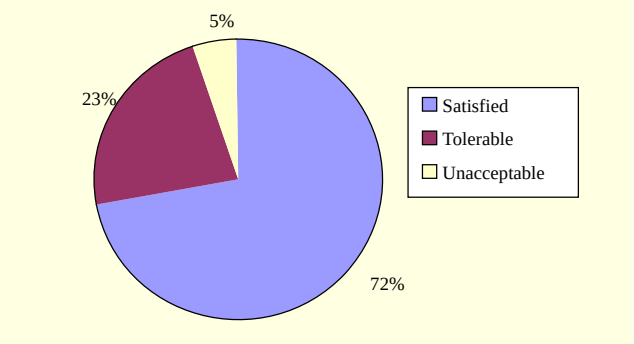


Fig. 9. Users' satisfaction survey.

The distinguishing characteristics of the three levels are as follows.

- Satisfactory: This level implies that the student considers the outcome of webpage adaptation to be satisfactory. The participant can obtain the desired information and content quickly using the mobile smartphone. Compared to ordinary PC browsing, it does not cause any discomfort for the user.
- Tolerable: This level implies that the student considers the outcome of the webpage adaptation to be acceptable; the errors are considered minor and do not hinder the student's overall understanding of the page. Compared to ordinary PC browsing, the user feels that there are some differences but that they are negligible.
- Unacceptable: This level implies that the student considers the adapted display to be significantly different from the original webpage and even confusing. The student cannot access the wanted information and content using the mobile smartphone. Compared to ordinary PC browsing, the user feels it is unsuitable.

The results of the questionnaire show that most testers find our adaptive conversion system to be satisfactory. As shown in Fig. 9, the volunteers rate the system as follows: 72% of the volunteers are satisfied with the adaptation, 23% of the volunteers find the adaptation tolerable, and 5% of the

Table 2. Statistics of conversion accuracy.

| Website | Number of pages | | | Accuracy |
|---|---|---|---|---|
| | Total | Accurate transformation | Incorrect transformation | |
| sina.com | 116 | 112 | 4 | 96.55% |
| jrj.com.cn | 98 | 92 | 6 | 93.88% |
| ifeng.com | 107 | 98 | 9 | 91.59% |
| 163.com | 107 | 98 | 9 | 91.59% |
| sohu.com | 103 | 94 | 9 | 91.26% |
| qq.com | 103 | 92 | 11 | 89.32% |
| huanqiu.com | 63 | 55 | 8 | 87.30% |
| xinhuanet.com | 93 | 80 | 13 | 86.02% |
| chinanews | 88 | 75 | 13 | 85.23% |
| hexun.com | 98 | 80 | 18 | 81.63% |
| caixun.com | 73 | 62 | 11 | 84.93% |
| xinmin.cn | 71 | 0 | 71 | 0.00% |
| Sum | 1120 | 938 | 182 | 83.75% |

volunteers are dissatisfied with the results.

## 6. Quantitative Evaluation

To verify the accuracy of the algorithm, we select 1,120 webpages as an experiment data set and choose 30 juniors as our test volunteers. Using manual observation to determine the content of the pages, a comparison to the results of our adaptive conversion method is made.

- **Case 1.** The result from manual observation is identical to the result of the conversion.
- **Case 2.**

$$\frac{|\text{Manual observation of result} - \text{Automatic transformation of result}|}{\text{Manual observation of result}} < 5\%$$

- **Case 3.** A program error occurs.

$$\text{Accuracy} = \frac{\text{Correct transformed number of pages}}{\text{Total number of pages}} \quad (6)$$

In Cases 1 and 2, the conversion is accurate. In Case 3, as well as in other cases not mentioned, an error occurs. According to the experiment results shown in Table 2, we find that the accuracy of the algorithm for some sites is up to 96%, while the average accuracy of all samples is 83.75%. We also note that the accuracy rate is zero for some individual sites, such as xinmin.cn. A statistical analysis shows that such sites are constituted by a large number of pictures and lack textual information, so our method is not applicable for these websites.

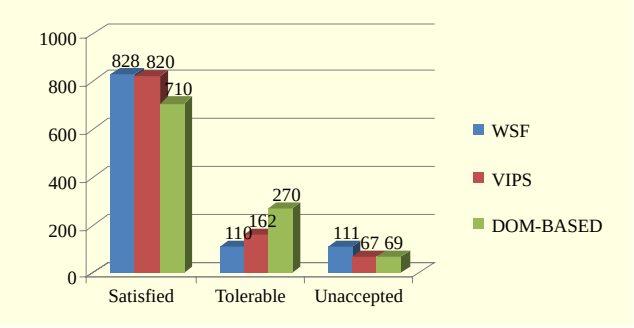


Fig. 10. Comparisons with other content extraction methods.

Table 3. Execution time comparisons before and after adaptive transformation

| No. | Webpage type | Execution time (seconds) | | Saved time ration |
|---|---|---|---|---|
| | | Source webpage | After transformation | |
| 1 | Content | 11.8 | 1.71 | 85.51% |
| 2 | Content | 12.3 | 1.86 | 84.88% |
| 3 | Content | 13.4 | 1.95 | 85.45% |
| 4 | Content | 12.8 | 1.92 | 85.00% |
| 5 | Index | 32.8 | 1.82 | 94.45% |
| 7 | Index | 35.6 | 1.62 | 95.45% |
| 8 | Index | 33.4 | 1.74 | 94.79% |
| 9 | Index | 37.9 | 1.69 | 95.54% |

## 7. Comparison with Other Methods of Web Content Extraction

For webpage content extraction, VIPS [14] and DOM-based [15] algorithms are compared with our method based on WSFs. Our experimental data set consists of the data presented in subsection III.6, with the exception of xinmin.cn. Similarly, we choose 30 juniors as our test volunteers. Using manual observation and auto transformation by different methods, we obtain the experiment results shown in Fig. 10. These results show that the performance of our WSF-based method matches that of the VIPS method but exceeds that of the DOM-based method.

## 8. System Execution Time Comparison

The system execution time is calculated as follows:

$$T_u = T_c + T_{process} + T_s, \quad (7)$$

where $T_c$ denotes the delay of users connected to the proxy, $T_s$ represents the delay of the application connected to the remote server, and $T_{process}$ is the application processing time. From (7), we can easily extract that $T_c$ and $T_s$ are affected by the network,

compared with $T_{process}$, $T_c$ and $T_s$ can ignore. Therefore, we only took account in $T_{process}$ in our experiment. In our experiments, under the GPRS network environment and the 10-kbps bandwidth, we use four index webpages and four content webpages, respectively, by mobile terminals, before and after adaptive transformation. The experiment results show that the time saved by users is more than 84% (see Table 3).

## IV. Conclusion

This paper presented an adaptive transformation method based on webpage semantic features to solve the problem small-screen mobile terminals have in accessing existing Web resources designed for large-screen personal computers. According to the text density and link density features of the webpage, we extracted the main content of the webpage via a content-based webpage transformation algorithm and an index-based webpage transformation algorithm, allowing the mobile terminals to adaptively access Internet services. Experiments showed that our method is not dependent on specific software and webpage templates, and it can effectively enhance Web content adaptation on small-screen terminals. In future work, we must improve the accuracy of the page type judgment algorithm and handle webpages whose main contents are pictures and videos as well as handle the interactive functions of webpages.

## References


[1] H. Alam and F. Rahman, "Web Document Manipulation for Small Screen Devices: A Review," *Proc. Int. Works. Web Document Anal.*, 2003, pp. 33-36.

[2] H. Lam and P. Baudisch, "Summary Thumbnails: Readable Overviews for Small Screen Web Browsers," *Proc. CHI*, Portland, OR, USA, Apr. 2005, pp. 681-690.

[3] S. Saha, M. Jamtgaard, and J. Villasenor, "Bringing the Wireless Internet to Mobile Devices," *Computer*, vol. 34, no. 6, 2001, pp. 54-58.

[4] S.J. Barnes and B. Corbitt, "Mobile Banking: Concept and Potential," *Int. J. Mobile Commun.*, vol. 1, no. 3, 2003, pp. 273-288.

[5] D. Cai et al., "Vips: A Vision Based Page Segmentation Algorithm," Technical Report MSR-TR-2003-79, Microsoft Research, 2003.

[6] S. Baluja, "Browsing on Small Screens: Recasting Web-Page Segmentation into an Efficient Machine Learning Framework," *Proc. 15th Int. Conf. World Wide Web*, Edinburgh, Scotland, May 23-26, 2006, pp. 33-42.

[7] D. Chakrabarti, R. Kumar, and K. Punera, "A Graph-Theoretic Approach to Webpage Segmentation," *Proc. 17th Int. Conf. World Wide Web*, Beijing, China, Apr. 21-25, 2008, pp. 377-386.

[8] P. Xiang, X. Yang, and Y. Shi, "Web Page Segmentation Based on Gestalt Theory," *Proc. IEEE Int. Conf. Multimedia Expo*, 2007, pp. 2253-2256.

[9] S.J.H. Yang et al., "Applying Semantic Segment Detection to Enhance Web Page Presentation on the Mobile Internet," *J. Inf. Sci. Eng.*, vol. 27, no. 2, 2011, pp. 697-713.

[10] J. Deng et al., "The Web Data Extracting and Application for Shop Online Based on Commodities Classified," *Comput. Intell. Syst.*, 2011, vol. 234, pp. 189-197.

[11] C. Kohlschütter and W. Nejdl, "A Densitometric Approach to Web Page Segmentation," *Proc. 17th ACM Conf. Inf. Knowl. Manag.*, Napa Valley, CA, USA, Oct. 26-30, 2008, pp. 1173-1182.

[12] R. Györödi et al., "Web Page Analysis Based on HTML DOM and Its Usage for Forum Statistics, Alerts and Geo Targeted Data Retrieval," *WSEAS Trans. Comput.*, vol. 9, no. 8, 2010, pp. 822-831.

[13] K. Vieira et al., "A Fast and Robust Method for Web Page Template Detection and Removal," *Proc. 15th ACM Int. Conf. Inf. Knowl. Manag.*, Nov. 06-11, 2006, pp. 258-267.

[14] D. Cai et al., "Extracting Content Structure for Web Pages Based on Visual Representation," *Web Technol. Appl.*, vol. 2642, 2003, pp. 406-417.

[15] S. Gupta et al., "DOM-Based Content Extraction of HTML Documents," *Proc. 12th Int. Conf. World Wide Web*, May 20-24, 2003, pp. 207-214.



**Hao Li** received his MS from Huazhong Normal University in 2010. He is now a doctoral student at the National Engineering Research Center for E-Learning of Huazhong Normal University. He has authored five published journal articles and conference papers. His main research interests include mobile applications and the Semantic Web.



**Qingtang Liu** received his PhD from Huazhong University of Science and Technology in 2005. He is now a professor, PhD candidate supervisor, and the dean of the College of Information and Journalism Communication at Huazhong Normal University, China. He has authored more than 20 published journal articles and conference papers. His main research interests include digital copyright protection, search engines, mobile applications, Web 2.0, and the Semantic Web.

**Min Hu** received her MS from Huazhong Normal University in 2011. She is now a doctoral student at the National Engineering Research Center for E-Learning of Huazhong Normal University. She has authored five published journal articles and conference papers. Her main research interests include knowledge services and the Semantic Web.

**Xiaoliang Zhu** received his BS from Hefei University of Technology in 1996 and received his MS and PhD degrees from the Department of Electronics and Information Engineering of Huazhong University of Science and Technology in 2003 and 2006, respectively. He is now a vice professor at the National Engineering Research Center for E-Learning of Huazhong Normal University. He has authored more than 10 published journal articles and conference papers. His main research interests include multimedia information processing and IPTV technology.