Knowl Inf Syst
DOI 10.1007/s10115-013-0687-x

REGULAR PAPER

A hybrid approach for content extraction with text density and visual importance of DOM nodes

Dandan Song · Fei Sun · Lejian Liao

Received: 22 December 2012 / Revised: 27 August 2013 / Accepted: 14 September 2013

Abstract Additional contents in web pages, such as navigation panels, advertisements, copyrights and disclaimer notices, are typically not related to the main subject and may hamper the performance of Web data mining. They are traditionally taken as noises and need to be removed properly. To achieve this, two intuitive and crucial kinds of information—the textual information and the visual information of web pages—is considered in this paper. Accordingly, Text Density and Visual Importance are defined for the Document Object Model (DOM) nodes of a web page. Furthermore, a content extraction method with these measured values is proposed. It is a fast, accurate and general method for extracting content from diverse web pages. And with the employment of DOM nodes, the original structure of the web page can be preserved. Evaluated with the CleanEval benchmark and with randomly selected pages from well-known Web sites, where various web domains and styles are tested, the effect of the method is demonstrated. The average F1-scores with our method were 8.7 % higher than the best scores among several alternative methods.

Keywords Content extraction · Text density · Visual importance

1 Introduction

With the fast development of the Internet, tremendous information is increasingly contained in large quantities of web pages. Researches on it always require *main content* (e.g., an article text) from the web to be gathered, processed and stored quickly and efficiently. However, the

D. Song · F. Sun · L. Liao (✉)
Beijing Engineering Research Center of High Volume Language Information Processing and Cloud Computing Application, Beijing Lab of Intelligent Information Technology, School of Computer Science and Technology, Beijing Institute of Technology, 100081 Beijing , China
e-mail: liaolj@bit.edu.cn

Present Address:
F. Sun
Institute of Computing Technology, Chinese Academy of Sciences, 100190 Beijing, China

Published online: 26 September 2013

main content in web pages is often accompanied by a large amount of additional content such as navigation menus, banner advertisements and copyright notices. Although such information is expected by Web site owners and benefits user browsing, it is unrelated to the topic of the web pages and not simple enough for computer programs to parse. This information can be treated as noise, which needs to be removed properly. So we define the main content as the informative part of a web page that is related to the topic of the web page and can provide the web readers some valuable information. And for some news web pages, the comments are also included in the main content as they are always informative as well.

Content extraction techniques to extract the main content and remove noise becomes essential for many real-world applications. For example, the performance can be improved for web mining and information retrieval from web pages, which are important and popular techniques for discovering useful information or knowledge. Furthermore, building text corpora was traditionally a very expensive and time-consuming process. By automatically downloading textual data from the web, extremely large corpora can be built in a short period, at relatively low cost. Therefore, the idea of *Web as Corpus* has been very attractive for many researchers in Natural Language Processing and related areas. In order to prepare web data for use as a corpus, ACL-SIGWAC held the first CleanEval competition during the summer of 2007 [26]. And as Pocket-sized devices with small screens such as tablet PCs and smart mobile phones have become ubiquitous, adapting web pages for small screens has become an increasingly important and challenging task [2,7]. Additionally, these techniques are also utilized for navigation page detection [22] and incremental microblog crawler [30].

However, extracting the main content from web pages has become more difficult and non-trivial. As early as 2005, Gibson et al. [15] estimated the noise to account for around 40–50 % of the data on the web and predicted correctly that this ratio would go on increasing. Moreover, web page layout has become much more complex than before, especially with the development and widespread use of the Cascading Style Sheets (CSS) technique. In our observation, most of the recent web pages use the style sheets and or tags for structural information to replace structural tags within a web page. Consequently, many early content extraction techniques failed to keep up with these changes and performed poorly, because particular HTML cues (e.g., <table>, <td> and fonts) that they used before are no longer included in recent web pages. Meanwhile, nowadays most additional web page content, especially advertisements (e.g., Google AdSense), is generated dynamically, which make template detection algorithms perform poorly. These wrapper generation algorithms may also break easily due to frequent changes in page structure.

In this paper, we propose a highly effective content extraction algorithm to extract main content from web pages. Our content extraction technique is based on the following two observations. The first one is that, in a typical web page, the noise is usually highly formatted and contain less text and brief sentences, whereas the content is commonly simply formatted and contain more text but far less hyperlinks than in the noise. We call this the *Textual Information*. The second one is that, the main content is always displayed in the central part of a web page, which makes it easy to be identified for human beings. We call this the *Visual Information*. Furthermore, the content is usually an integral part of a web page and maintains the integrity of the structure, i.e., belongs to an ancestor node in the DOM tree.

With the textual information, we propose two measures for the evaluation of the textual importance of tags in web pages: *Text Density* and *Composite Text Density*. Once an HTML document is parsed and represented by a DOM tree, we calculate the text density for each node. Higher text density implies the node is more likely to represent a tag with content text within the web page. In the case of noise, the opposite applies. Afterward, we extend the Text Density to the Composite Text Density by adding statistical information about hyperlinks.

Furthermore, with the visual information, we defined a visual measure for the evaluation of tags in web pages: *Visual Importance*. It is calculated by considering relative displaying positions and sizes of tag nodes on web pages. Then, we combine the Visual Importance with the Composite Text Density and proposed a *Hybrid Text Density*. Additionally, to solve the problem of losing low text density nodes in content, a tailored technique called *DensitySum* was designed to extract integral content.

The results show that it is a fast, accurate and general content extraction algorithm, which outperforms many current content extraction algorithms on large and varied data sets. One difference between our approach and current methods is that we make no assumptions about the specific structure of an input web page, nor do we look for particular HTML cues. Another is that we can not only extract the main content, but also preserve the original structure of the input pages since all operations are performed on their DOM trees. Most existing approaches remove all tags from the HTML document and just output the text of the contents, which are not user-friendly, and cannot be used to extract structured data from the web pages. In contrast, our method can output cleaned HTML documents instead of unformatted text.

The rest of the paper is organized as follows: after briefly reviewing related work in Sect. 2, we propose our method in Sect. 3, including definitions of text density and composite text density, definition of visual importance and hybrid text density, as well as how to choose the threshold and the algorithm to extract content. Then, we describe our evaluation setup and compare the performance of our approach with other content extraction methods, and discuss the results in Sect. 4. Finally, we offer our conclusions and plans for future research.

2 Related work

The term *Content Extraction* (CE) was introduced by Rahman et al. [29]. In the last decade, extraction of content from web pages has been studied intensively and numerous methods have been developed.

In the early days of content extraction, some handcrafted web scrapers (such as NoDoSE [1] or XWRAP [24]) extracted article text embedded in a common template from web pages by looking for some HTML cues, using regular expressions. They were written in a traditional programming language or with some specialized tools designed for content extraction. The biggest advantage of these methods was their accuracy. Obviously, the disadvantage lies in the fact that different regular expressions need to be manually created for each Web site. Still, even individual Web sites employ multiple structures; furthermore, these Web sites may also change structures or layouts over time. All the above situations mean such approaches require constant updating.

Kushmerick [21] and Davison [8] proposed machine learning mechanisms to recognize banner advertisements, redundant and irrelevant links in web pages. And Bu et al. [4] proposed a statistics-based approach that integrates the concept of fuzzy association rules (FAR) with that of sliding window (SW) to efficiently extract the main text content from web pages. However, these techniques cannot be put to general use because they require a large set of manual-labeled training data and domain knowledge to generate classification rules.

There are also Template Detection (TD) algorithms proposed [3,6,19,23,34] in which collections of documents based on the same template are used to learn a common structure. Bar-Yossef and Rajagopalan [3] presented an approach to automatically detect templates from the largest *pagelet*, i.e., self-contained regions in a web page. Lin and Ho [23] partitioned a page with <table> tags and identified redundant blocks using an entropy measure over a set of word-based features. In order to improve the performance of web page clustering and

classification, Yi et al. [34] introduced a *site style tree* (SST) structure that labels DOM nodes with similar styles across pages as uninformative. Chen et al. combined template detection and removal with the index building process in large-scale search engines to increase accuracy and speed. They segmented pages into blocks and clustered them based upon their styles and positions; then similar clusters among different pages were identified as part of the template [6].

In general, template detection algorithms identify the content by removing identical parts found in all web pages. This is an accurate approach but has been found to be too burdensome. The reason is that models need to be built for each Web site. This means pages in each site should share the same template. Furthermore, these methods often incorrectly assume that uninformative segments are largely repeated across pages (this is clearly not the case for varying text advertisements or lists of related articles), and any updates of the layout or structure may result in the template's failure.

Conversely, in the CleanEval shared task, only a few pages are available from the same site, thus requiring a more general approach. The winner of the CleanEval task split pages by their tags into a sequence of blocks and then labeled each block as "content" or "noise" using conditional random fields with a number of block-level features [26].

There are also sets of content extraction approaches based on statistical information of web pages. Finn et al. [13] introduced the Body Text Extraction (BTE) algorithm to improve the accuracy of the content's classifier for digital libraries. They interpreted an HTML document as a sequence of word and tag tokens, and then extracted content by identifying a single, continuous region, which contains the most words and the least HTML tags. To overcome the restriction of BTE in discovering only a single continuous block of text, Pinto et al. [28] extended this method to construct Document Slope Curves (DSC), in which a windowing technique is used to locate document regions in which word tokens are more frequent than tag tokens. They used this technique to improve performance and efficiency for answering questions with web data in their QuASM system.

Mantratzis et al. [25] presented an approach named Link Quota Filter (LQF) to identify link lists and navigation elements by identifying DOM elements, which have a high ratio of text residing in hyperlink anchors. It can be applied to content extraction by removing the resulting link blocks from the document. The drawback of this method is that it relies on structure elements, and it can only identify hyperlink-type noise.

Debnath et al. [9,10] proposed the FeatureExtractor (FE) and KFeatureExtractor (KFE) techniques based on block segmentation of the HTML document. Each block is analyzed for particular features such as the amount of text, the presence of images and script code. Content text is extracted by selecting blocks that correspond best to a desired feature, e.g., the presence of most text.

The Content Code Blurring (CCB) algorithm was introduced by Gottron [17]. Content regions are detected in homogeneously formatted source code character sequences.

Weninger et al. introduced the Content Extraction via Tag Ratios (CETR) algorithm, a method to extract content text from diverse web pages using the HTML document's tag ratios [33]. The approach computes tag ratios on a line-by-line basis and then clusters the resulting histogram into content and noise areas. This is a simple and efficient algorithm, however vulnerable to the page's source code style changes.

Kohlschütter et al. [20] developed a simple, yet effective technique to classify individual text elements from a web page. They analyzed a small set of shallow text features, which were theoretically grounded by stochastic text generation processes from Quantitative Linguistics, for boilerplate detection. Their study provides theoretical support for the method in this paper.

Gupta et al. [18] attempted to combine different heuristics into one system called the *Crunch* framework. They demonstrated that a well-chosen combination of different content extraction algorithms can provide better results than a single approach on its own. Since Crunch, several new content extraction algorithms have been developed. Then, Gottron [16] developed *CombineE* framework, a ensemble method, which made it easier to configure ensembles of content extraction algorithms. In a most recent work, Peters and Lecocq [27] explore a machine learning approach to content extraction that combines diverse feature sets and methods.

A different approach for content extraction is the vision-based page segmentation (VIPS) technique that was introduced by Cai et al. [5] to divide a web page into a tree, where the nodes are visually grouped blocks. Based on the VIPS method, Song et al. [31] presented an approach to rank block importance for web pages through learning algorithms using spatial features (such as position and size) and content features (such as the number of images and links), and Fernandes et al. [12] developed another way to compute block importance of a web page by means of assigning weights to classes of structurally similar blocks. But the visual information alone is not enough to extract contents accurately. Fumarola et al. [14] proposed a hybrid approach to extract general list on web pages. However, as specific hints of lists are used, its application is confined.

3 Methods

In this section, based on the basic DOM tree structure and an example, we firstly give out the definitions of *Text Density* for each node in a DOM tree, and then extend the text density to *Composite Text Density* by adding statistical information about hyperlinks. These are the two measures based on the textual information. Afterward, we propose the definition of *Visual Importance*, which is based on the visual information, and later the *Hybrid Text Density*. Furthermore, we design a tailored technique—*DensitySum* to extract integral content from a web page.

3.1 Basics

3.1.1 An example

Let us take the news article from The Financial Times[1] shown in Fig. 1 as an example. This page is typical of the web: the banner, navigation and advertisements take up about half space on the page, while the content of the page is confined to a relatively small space.

3.1.2 DOM tree

Document Object Model (DOM) [32] is a standardized, platform- and language-independent interface for accessing and updating content, structure and style of documents. Each HTML page corresponds to a DOM tree where tags are internal nodes and the detailed text and images are leaf nodes.

[1] http://www.ft.com.

Fig. 1 The Financial Times web page article

Example 3.1 Below is a brief segment of HTML code from Fig. 1.

```
1. <div class="main">
2.   <div class="article">
3.     <div class="story-header">
4.       Lunch with the FT: Biz Stone</div>
5.     <div class="story-body">
6.       Though the value of the company was
7.       <a>recently estimated at $3.7bn</a>
8. </div></div></div>
```

Example 3.1 shows a segment of HTML code from Fig. 1, and Fig. 2 shows the DOM tree of Example 3.1.

Our study of HTML web pages begins from the <body> tag since all the viewable parts in a web page are within the scope of <body>.

3.2 Text density

To extract content from a web page, we take advantage of typical textual features of content and noise. It was found that the noise in web pages is usually highly formatted and contains less text and brief sentences. On the other hand, the content is commonly lengthy and simply formatted. The example in Fig. 1 supports this observation.

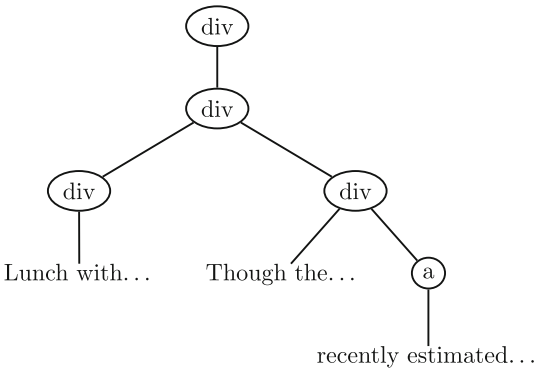

Fig. 2 The DOM tree of Example 3.1

Once an HTML document has been parsed and is represented by a DOM tree, the number of characters and tags that each node contains can be figured out. Then, such statistical information can be added to the node.

- *CharNumber*: number of all characters in its subtree.
- *TagNumber*: number of all tags in its subtree.

Furthermore, we can compute a ratio of the number of characters to the number of tags per node. Now, we define the *Text Density*, the basis of our method, as follows:

Definition 3.1 If i is a tag (corresponding to an element node in DOM) in a web page, then the tag i's Text Density (TD_i) is the ratio of its CharNumber to its TagNumber:

$$TD_i = \frac{C_i}{T_i} \tag{1}$$

where C_i is the number of all characters under i, T_i is the number of all tags under i. Note that if T_i is 0, T_i is set to be 1, which is like a pseudo-count.

TD_i is a measure of the density of each node's text in a web page. It assigns high values for nodes that commonly contain long and simply formatted text and low values for highly formatted nodes containing less, brief text. It is useful for determining whether a part of a web page is meaningful or not. Clearly, content in a web page will be assigned relatively high density.

Before performing the computation, *script*, *comment* and *style* tags are removed from the DOM tree because such information is not visible and would likely skew the results if included in computation.

The *ComputeDensity* algorithm is described as Algorithm 1 where N is a DOM node being computed.

Computing the text density is a recursive task as evident from the simplicity of Algorithm 1. Example 3.2 below shows the text density for each node of Example 3.1.

Example 3.2 The text density for the five tags in Example 3.1 is computed as follows:

```
1. <div "main">: Chars=91, Tags=4, Density=22.75
2. <div "article">: Chars=91, Tags=3, Density=30.33
3. <div "story-header">: Chars=28, Tags=1, Density=28
4. <div "story-body">: Chars=63, Tags=1, Density=63
5. <a>: Chars=28, Tags=1, Density=28
```

Figure 3 shows the resulting density histogram for the page in Fig. 1. It can be seen that there are nodes with a relatively high text density. Intuitively, we can take the high text density portion as the web page's content.

Algorithm 1 Pseudocode of ComputeDensity(N).

```
INPUT: N
OUTPUT: N.Density
for child node C of N do
    ComputeDensity(C)
end for
N.CharNumber ← CountChar(N)
N.TagNumber ← CountTag(N)
if N.TagNumber == 0 then
    N.TagNumber ← 1
end if
N.Density ← N.CharNumber/N.TagNumber
```

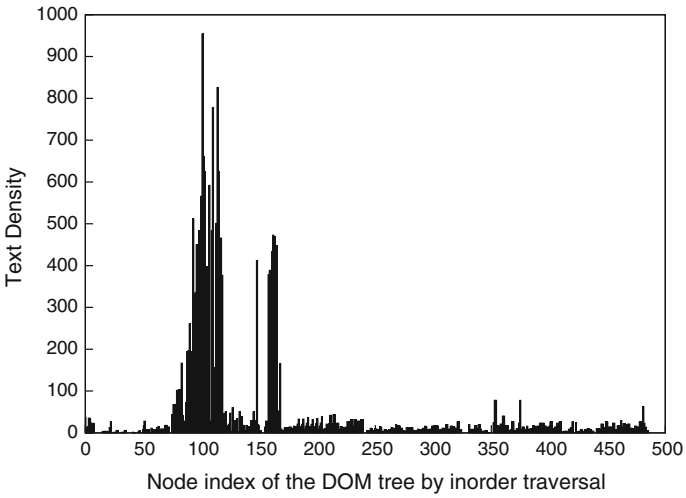

Fig. 3 Text density for each node from the FT web page

3.3 Composite text density

With further study, we find that most of the noise in the web pages consists of hyperlinks. The example web page in Fig. 1 also supports this observation.

Based on the above discussion, we calculate additional statistical information per node as below:

- *LinkCharNumber*: number of all hyperlink characters in its subtree.
- *LinkTagNumber*: number of all hyperlink tags in its subtree.

According to the four pieces of statistical information mentioned above, we redefine the Text Density. In order to distinguish it from the Text Density defined above, we call it *Composite Text Density*.

Definition 3.2 If i is a tag (corresponding to an element node in DOM) in a web page, then its Composite Text Density (CTD_i) is as follows:

$$CTD_i = \frac{C_i}{T_i} \log_{\ln\left(\frac{C_i}{\neg LC_i} LC_i + \frac{LC_b}{C_b} C_i + e\right)} \left(\frac{C_i}{LC_i} \frac{T_i}{LT_i}\right) \tag{2}$$

where C_i is the number of all characters under i, T_i is the number of all tags under i, LC_i is the number of all hyperlink characters under i, $\neg LC_i$ is the number of all non-hyperlink

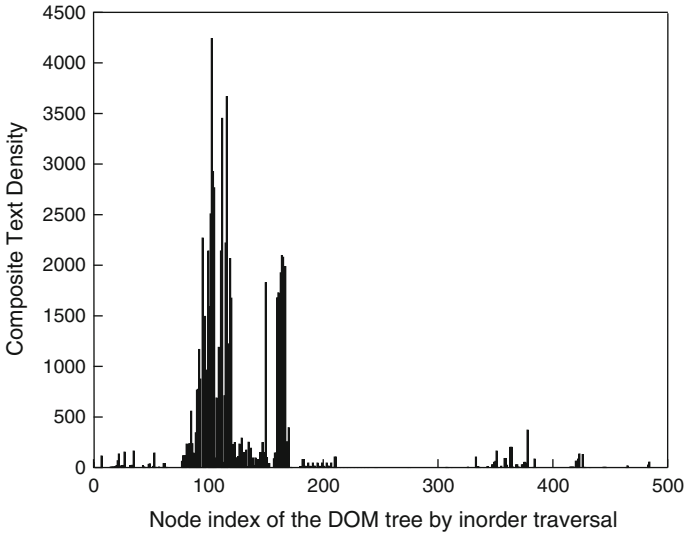

Fig. 4 Composite text density for each node from the FT web page

characters under i, LT_i is the number of all hyperlink tags under i, LC_b is the number of all hyperlink characters under the <body> tag, and C_b is the number of all characters under the <body> tag. Note that when denominators of the formula are 0, set them to 1.

In Definition 3.2, $\frac{C_i}{LC_i}$ is the inverse of the proportion of hyperlink texts in i; accordingly, $\frac{T_i}{LT_i}$ is the inverse of the proportion of hyperlink tags. When tag i contains numerous non-hyperlink characters and few hyperlinks, they will assign high values to it, and vice versa. Meanwhile, $\frac{C_i}{\neg LC_i} LC_i$ would get a low value in this case, and high otherwise. The role of $\frac{LC_b}{C_b} C_i$ is to maintain balance by preventing nodes containing lengthy and homogeneously formatted text from getting extremely high values, or nodes that contain brief text (e.g., news headlines or one sentence paragraphs) from getting extremely low values. The ln calculation is to reduce the magnitude of the base, thus make the value after log more distinguishable. And the added e is to ensure the value after ln, which is the base of the log calculation, is greater than 0.

We argue that a node with too many hyperlinks and less text is less important, thus getting a low-density value; and a node that contains much non-hyperlink text and few hyperlinks is more important, and receives a high density value. Clearly, a node's density will be zero if there are only hyperlinks in its subtree. In another extreme case, all tags of a web page which have no hyperlinks will get an infinite density. Thus, we would classify such a page as not containing noise.

Compared to Fig. 3, Composite Text Density, shown in Fig. 4, is better suited for classification because of the more obvious differences between sections.

3.4 Visual importance and hybrid text density

For human beings, intuitively, the main content of a web page traditionally occupies the central part of the screen. Inspired by this, the visual information—*Visual Importance*—is defined and incorporated into the text density, which is defined as *Hybrid Text Density*.

3.4.1 Visual importance

Relative displaying positions and sizes of DOM nodes are considered as useful visual information for content extraction. As each DOM node will be rendered to a rectangle on the web page, its size and the coordinates of its central point can be easily attained with a web development kit. Here, the horizontal data are employed, as we assume that the main content is more likely to be located in the horizontally central part of a page.

For each leaf node in the DOM tree structure (which corresponds to an innermost tag encompassing text only), the *Visual Importance* (VI) considers its relative displaying size and location and is calculated as follows:

- Firstly, all the locations and sizes of DOM nodes are gathered. Then, in the DOM tree structure, the node with a longest horizontal span under the root node is selected (denoted as $Node_L$), and its sided boundaries, supposing x_{L1} and x_{L2} ($x_{L2} > x_{L1}$) on the horizontal dimension are identified.
- The area under $Node_L$ is intuitively taken as the visually important area. The visual importance distribution is empirically illustrated by a standard normal distribution $N(0, 1)$. Specifically, points of x_{L1} and x_{L2} are mapped to points -1 and $+1$ on the variable axis of the standard normal distribution function, respectively.
- Based on the above mapping, a normal distribution $N(\mu, \sigma^2)$ for the visual importance description of the given page is constructed. As a result, the mean of the distribution is assigned to be $\mu = \frac{x_{L1}+x_{L2}}{2}$, with the variance to be $\sigma = \frac{x_{L2}-x_{L1}}{2}$. And the consequential probability density function is $f(x) \sim N(\frac{x_{L1}+x_{L2}}{2}, \frac{x_{L2}-x_{L1}}{2}^2)$.
- As each informative leaf node in the DOM tree occupies a horizontal space under the normal distribution in the web page, its Visual Importance (VI) is defined as the convolution of the distribution under its scope. So if a leaf node is hidden thus occupies no space, its Visual Importance will be 0 as we think it is not important.

Definition 3.3 If i is a leaf node in the DOM tree, and the horizontal coordinates of its displaying boundaries are x_1 and x_2, then its *Visual Importance* (VI_i) is as follows:

$$VI_i = \int_{x_1}^{x_2} f(x)\mathrm{d}x \tag{3}$$

where $f(x)$ is the probability density function $f(x) \sim N(\frac{x_{L1}+x_{L2}}{2}, \frac{x_{L2}-x_{L1}}{2}^2)$, as x_{L1} and x_{L2} are horizontal boundaries of the node with a largest horizontal span under the root node. The calculation is illustrated in Fig. 5.

3.4.2 Hybrid text density

To combine textual and visual information for DOM nodes, the measure of visual important is incorporated into the Composite Text Density, redefined as *Hybrid Text Density*.

If i is a *leaf* node in the DOM tree, then its CharNumber (C_i) is adjusted to Hybrid CharNumber (HC_i) using the visual importance value as a weight, as $HC_i = VI_i * C_i$. For other tag nodes, the Hybrid CharNumber is defined as the sum of all its sons' Hybrid CharNumbers.

Consequently, the Hybrid Text Density is defined as follows:

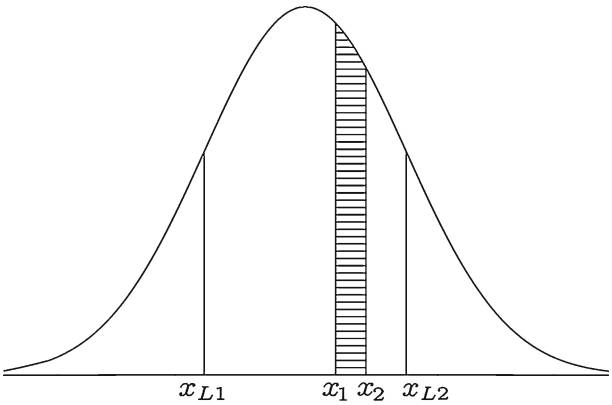

Fig. 5 Illustration of visual importance calculation for a DOM node with horizontal displaying boundaries x_1 and x_2

Definition 3.4 If i is a node in the DOM tree, then its Hybrid Text Density (HTD_i) is:

$$HTD_i = \frac{HC_i}{T_i} \log_{\ln\left(\frac{HC_i}{\neg LC_i} LC_i + \frac{LC_b}{HC_b} HC_i + e\right)} \left(\frac{HC_i}{LC_i} \frac{T_i}{LT_i}\right) \tag{4}$$

in which all the appearances of CharNumber in Eq. 2 are substituted by the Hybrid CharNumber. Specifically, C_i and C_b for node i and the <body> tag are, respectively, changed to HC_i and HC_b, while the remaining of Eq. 4 is exactly the same with Eq. 2.

It must be noticed that, previously in the definitions of Text Density and Composite Text Density, T_i is set to be 1 when it is 0. However, in the above definition of the Hybrid Text Density, as the Visual Importance value VI is in the interval of [0, 1], the derived Hybrid CharNumber is probably less than 1. To avoid the Hybrid Text Density being negative, an adaption is designed here: if T_i is 0, it is set to be HC_i/C_i, where C_i and HC_i are the initial and the Hybrid CharNumbers, respectively.

With the Hybrid Text Density, the resulted density histogram for the FT page is shown in Fig. 6.

Unless otherwise specified, we use Text Density to refer to the initial Text Density, the Composite Text Density and the Hybrid Text Density. And these calculation methods are discussed further in Sect. 4.

3.5 Content extraction

In this subsection, we describe the technique used to extract the content. The idea behind this approach is to determine a threshold t, which divides nodes into content or noise sections. The problem then becomes a matter of finding the best value of the threshold and a way to extract the content completely.

3.5.1 DensitySum

It is easy to see that some nodes' text density is abnormally different from the surrounding nodes in Figs. 3, 4 and 6. For instance, pictures, hyperlinks, the byline or dateline of news articles, or very short paragraphs in the main article and references of an article may have abnormally low text density; conversely, some noise nodes (e.g., copyright or disclaimer

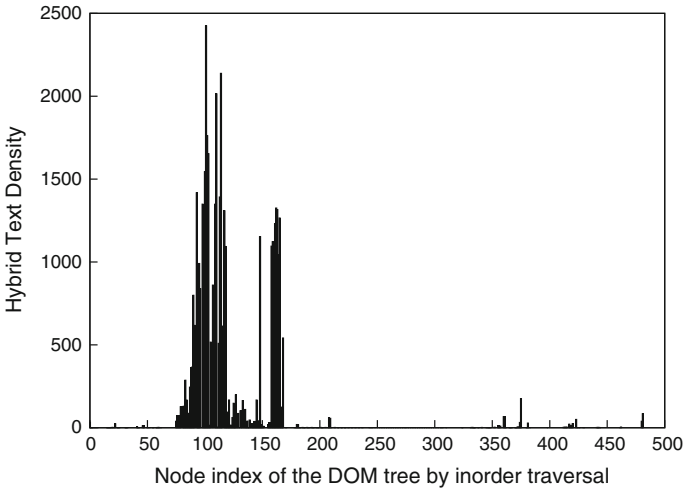

Fig. 6 Hybrid text density for each node from the FT web page

information) may get an abnormally high text density. Therefore, many content nodes may be lost if we simply label a node as content or noise just according to the threshold, and some noise nodes may be retained.

It is known that Data Smoothing can help with this problem to a certain extent by smoothing outlying peaks and valleys, increasing cohesiveness within sections and differences between sections. In general, although data smoothing can achieve good results, it may still lose some low-density content nodes; meanwhile, some noise nodes may be retained because of text density increase.

To solve this problem, we propose a technique called *DensitySum*. It is based on the observation that a content block of a web page belongs to an ancestor node in the structure of DOM; and as mentioned before, the text density of content nodes is much higher than that of noise nodes. Therefore, the content block (i.e., a node in DOM) will get a peak value if we add up its children's text densities. Here, we define DensitySum as below:

Definition 3.5 If N is a tag (corresponding to an element node in DOM) in a web page and i is a child of N, then N's DensitySum is the sum of all its children's text densities:

$$DensitySum_N = \sum_{i \in C} TextDensity_i \tag{5}$$

where C is the set of N's children and $TextDensity_i$ is the text density of tag i.

Note that $TextDensity$ here is just a general term; we use Text Density, Composite Text Density or Hybrid Text Density in practice.

3.5.2 Threshold

For a simple case, if there exists just one content block, we can identify the content by looking for the node under the <body> tag with the maximum DensitySum, then mark it as content. Afterward, we extract the content by keeping its ancestors and subtrees. However, in many cases, the web page contains more than one content block. To resolve this problem, the algorithm first finds the maximum DensitySum tag in the whole page, without thresholding.

Next, we set the minimum text density of the node in the path from the maximum DensitySum tag to <body> tag as threshold. Then, for each node in the DOM tree whose the text density is greater than the threshold, we apply the same method described above to extract content.

The simple process of content extraction using DensitySum can be seen in Algorithm 2.

Algorithm 2 Pseudocode of ExtractContent(N)

```
1: INPUT: N
2: if N.TextDensity >= threshold then
3:     T ← FindMaxDensitySumTag(N)
4:     MarkContent(T)
5:     for child node C in N do
6:         ExtractContent(C)
7:     end for
8: end if
```

4 Experimental results

In this section, we describe experiments on real-world data from various Web sites to demonstrate the effectiveness of our method. Methodology designed for our experiments is described first. We then present and discuss our experimental results.

4.1 Methodology

4.1.1 Data set

In our experiments, we use data from two sources: (1) development and evaluation data sets from the CleanEval competition and (2) the data sets we gathered from several Web sites.

CleanEval: CleanEval is a shared task and competitive evaluation on the topic of cleaning arbitrary web pages [26]. Besides extracting content, the original CleanEval competition also asked participants to annotate the structure of the web pages: identify lists, paragraphs and headers. In this paper, we just focus on extracting content from arbitrary web pages. This data set includes four divisions: a development set and an evaluation set in both English and Chinese languages which are all hand-labeled. It is a diverse data set, only a few pages are used from each site, and the sites have various styles and structures. So this data set can test the generalization ability of the approach.

However, we found some errors in the data of CleanEval's gold standard in the experiment, for instance, garbled characters or texts clearly not part of the content. Therefore, we manually extracted the main content of these pages using a web browser and saved it into text files (gold standard) in UTF-8 (which can be obtained from the authors' Web site[2]). In our experiments, we only use the English data set and do not distinguish between development and evaluation documents since our approach does not require training.

Our Data Set: In order to evaluate our methods on varied sources, we produced a data set (which can also be obtained from the authors' Web site). This data set is separated into two non-overlapping sets. (1) The *Big 5*: Ars Technica, BBC, Yahoo!, New York Times, Wikipedia

[2] http://disnet.cs.bit.edu.cn/.

and (2) the *Chaos* data set chosen randomly from Google News and the best-known blog platforms such as WordPress and Blogger. For our purposes, we arbitrarily selected 100 pages from each of the *Big 5* and 200 pages from *Chaos*. All these pages' contents were labeled manually using a web browser and saved as UTF-8 text files to serve as the gold standard.

4.1.2 Performance metrics

Standard metrics were used to evaluate and compare the performance of different approaches. Specifically, precision, recall and F_1-scores were calculated by comparing the results of each method against the hand-labeled gold standard. Let a be the text in the extraction result and b be the text in the gold standard. Precision, recall and F_1-scores then follow from:

$$P = \frac{LCS(a,b).length}{a.length}, \quad R = \frac{LCS(a,b).length}{b.length} \tag{6}$$

$$F_1 = \frac{2 \times P \times R}{P + R} \tag{7}$$

where $LCS(a, b)$ is the longest common subsequence between a and b. It is important to note that every word in the document is considered to be distinct even if two words are lexically the same. However, other methods, such as CETR, treated a and b as a bag of words, i.e., two words are considered the same if they are lexically the same. The bag of words measurement is more lenient, thus the scores of these methods may be inflated.

CleanEval uses a different metric to evaluate the participants' performance. The scoring method is based on the *Levenshtein Distance* from the output of an extraction algorithm to the gold standard text (the number of insertions and removals of words necessary to align the gold standard text with the output of the extraction algorithm; substitutions are not allowed). The full formula is as follows:

$$Score(a,b) = 1 - \frac{distance(a,b)}{alignmentLength(a,b)} \tag{8}$$

where $alignmentLength(a, b)$ is the number of operations (insert, remove or align) required to align two word sequences. The Levenshtein distance is relatively expensive to compute, taking $O(|a| \times |b|)$ time, which can be prohibitively large when $|a|$ and/or $|b|$ are sufficiently large. We find that our data sets frequently include documents which are that large (i.e., size greater than 10,000 words). CleanEval's scoring script takes an extremely long time to return results, or even an "out of memory" error.

CleanEval's metric can be transformed into the following formula:

$$Score(a,b) = \frac{LCS(a,b).length}{a.length + b.length - LCS(a,b).length} \tag{9}$$

Therefore, we implemented a test program using *Longest Common Subsequence* instead. In our experiments, it performed very well, and is fast and robust.

4.1.3 Implementation details

Four distinct algorithms are implemented. The first is the Content Extraction via Text Density with DensitySum method, hereafter referred to as CETD-DS. The second is the Content Extraction via Composite Text Density with DensitySum method, hereafter referred to as CECTD-DS. To demonstrate the effect of the DensitySum method, a comparative Composite Text Density with Data Smoothing method, hereafter referred to as CECTD-S with

Table 1 Results for CETD-DS on various domains

Source	Precision (%)	Recall (%)	F_1 (%)	Score (%)
CleanEval-Eng	92.96	94.52	93.73	88.96
NYTimes	98.38	95.84	97.09	94.42
Yahoo!	83.16	85.90	84.51	72.84
Wikipedia	98.32	97.22	97.77	95.76
BBC	84.39	95.21	89.48	80.66
Ars Technica	97.81	98.85	98.33	96.71
Chaos	92.23	94.99	93.59	88.76

simply weighted averaging with sibling nodes, is also implemented. The last one is the Content Extraction via Hybrid Text Density with DensitySum method, hereafter referred to as CEHTD-DS, with the visual importance information incorporated. We use Content Extraction via Text Density (CETD) to refer to these algorithms collectively.

The program is implemented based on the WebKit[3] Kernel in C++. Before text densities are computed, the algorithm removes invisible parts of an HTML document: *scripts*, *style definitions* and *comments*. Given their similar roles in a web page, buttons and drop-down lists are treated as hyperlinks.

4.1.4 Alternative approaches

In order to evaluate the performance of our methods, we compare them with several other content extraction algorithms.

Several other algorithms (BTE, DSC, FE, KFE, LQ, CCB) described in Sect. 2 have been implemented in Java for the Combine*E* framework [16]. In addition, CETR was also implemented in Java [33]. Evaluation was performed by providing each HTML document as input to each algorithm and collecting the results.

4.2 Results

Comparative experiments are conducted within our method and with alternative approaches. All results are collected by calculating the average of each metrics over all samples.

4.2.1 Our method

Table 1 presents the results of the Content Extraction via Text Density with DensitySum (CETD-DS) method when given the task of extracting contents from the CleanEval, *Big 5* and the *Chaos* data sets.

Table 2 presents the results of Content Extraction via Composite Text Density with DensitySum (CECTD-DS) method.

Tables 1 and 2 clearly show that these two methods perform very well, especially CECTD-DS, which performs better than CETD-DS on the broader corpora. This shows that Composite Text Density is more suitable than Text Density as a measure of the importance of a tag in web pages.

[3] http://webkit.org/.

Table 2 Results for CECTD-DS on various domains

Source	Precision (%)	Recall (%)	F_1 (%)	Score (%)
CleanEval-Eng	95.87	97.15	96.51	93.87
NYTimes	99.69	98.16	98.92	97.86
Yahoo!	84.59	93.99	89.04	80.71
Wikipedia	98.25	92.77	95.43	91.49
BBC	86.15	97.95	91.67	84.44
Ars Technica	98.04	99.51	98.76	97.57
Chaos	96.21	96.10	96.15	93.47

Table 3 Results for CECTD-S on various domains

Source	Precision (%)	Recall (%)	F_1 (%)	Score (%)
CleanEval-Eng	90.35	92.60	91.46	87.24
NYTimes	96.72	96.56	96.64	94.41
Yahoo!	80.33	93.34	86.35	76.16
Wikipedia	98.02	97.61	97.81	95.75
BBC	82.55	93.77	87.80	79.65
Ars Technica	94.61	93.56	94.08	91.65
Chaos	89.64	92.86	91.22	86.17

Interestingly, the CETD-DS outperforms CECTD-DS for Wikipedia, especially in recall. The reason for this phenomenon is the high density of in-text hyperlinks and low noise in Wikipedia pages. For the Composite Text Density method, these hyperlink tags in contents may assign their ancestor tag a low-density value, even lower than the threshold. However, the Text Density method could be unaffected.

Table 3 presents the results of Content Extraction via Composite Text Density with Smoothing (CECTD-S) method.

Clearly, Tables 2 and 3 show that CECTD-DS performs far better than CECTD-S for most data sets, again except the Wikipedia site. They demonstrate that DensitySum deals more effectively than data smoothing with the situation where content nodes' text density is abnormally lower than the threshold, and noisy nodes' text density is abnormally higher than threshold.

These results show that the CECTD-DS performs far better than CETD-DS and CECTD-S. It is stimulating that the CleanEval scores of them are higher than the winner of CleanEval competition, which only scored 84.1 % on the English data set [26].

The precision of the above three methods is relatively lower when applied to Yahoo! and BBC (compared with other sources). For Yahoo!, it is probably because its web pages contain user comments after each article, and these comments' structure hierarchies are very deep and separate from the content. That is why our method cannot extract the whole comments block. It is easy to see more precise results from sources such as Ars Technica, which hides comments by default, and NYTimes, which does not accept comments at all. As for BBC, the reason is that there are hidden disclaimers at the bottom of each page, with very long text. These disclaimers, which should not be contents because they are not visible in the browser, are included in the results.

It is exciting to see that the above problems for Yahoo! and BBC are solved by incorporating visual information into the content extraction, demonstrated by their enhanced precision, F_1

Table 4 Results for CEHTD-DS on various domains

Source	Precision (%)	Recall (%)	F_1 (%)	Score (%)
CleanEval-Eng	94.97	94.07	94.52	90.61
NYTimes	99.72	95.96	97.80	95.69
Yahoo!	91.99	88.59	90.26	81.93
Wikipedia	96.58	90.41	93.39	87.86
BBC	95.55	96.46	96.00	92.29
Ars Technica	98.12	98.83	98.48	97.01
Chaos	94.74	96.42	95.57	92.05

Table 5 F_1-measures (%) for each algorithm in each source

Algorithm	CE	NYT	Yahoo	Wiki	BBC	Ars	Chaos	Ave.
BTE	92.22	76.23	69.64	82.74	80.73	80.44	83.78	80.83
CCB	86.24	72.03	62.46	67.82	67.72	76.92	75.47	72.66
DSC	74.07	91.68	83.23	48.62	83.76	93.09	86.79	80.17
FE	17.56	6.98	9.41	2.91	7.15	0.002	11.46	8.97
KFE	74.13	72.56	62.61	55.59	64.43	82.03	69.78	68.73
LQF	91.23	93.42	75.40	79.85	83.93	93.15	88.01	86.42
CETR	88.59	87.80	73.27	82.30	76.76	88.16	82.63	82.78
CETD-DS	93.73	97.09	84.51	97.77	89.48	98.33	93.59	93.50
CECTD-S	91.46	96.64	86.35	**97.81**	87.80	94.08	91.22	92.24
CECTD-DS	**96.51**	**98.92**	89.04	95.43	91.67	**98.77**	**96.15**	**95.21**
CEHTD-DS	94.52	97.80	**90.26**	93.39	**96.00**	98.48	95.57	95.15

The best scores are marked in bold
CE CleanEval-Eng, *NYT* NYTimes, *Yahoo* Yahoo!, *Wiki* Wikipedia, *Ars* Ars Technica, *Ave.* Average

and score values as shown in Table 4, which lists the results of the Content Extraction via Hybrid Text Density with DensitySum (CEHTD-DS). Additionally, the precision of NYtimes and Ars Technica is also enhanced. This demonstrates that the visual importance is a valuable information, especially for the pages whose contents are obviously displayed. However, the Recall values for these experimental Web sites are slightly reduced, except the Chaos data set. The explanation for this phenomena is that by using both the textual and visual information together, the criteria for the content extraction becomes stricter; thus, fewer but more trustable results are attained. Therefore, if extraction results are expected to be more precise, CEHTD-DS is more suitable than CECTD-DS. Without exception, this CEHTD-DS method is not suitable for the Wikipedia web pages, as most of their DOM nodes occupies a small space of the web page. Furthermore, the performance for CleanEval is slightly decreased, as there are many pages structured unconventionally or even strangely.

4.2.2 Methods comparison

In order to show the effectiveness of our method, we compare the above performance with the alternative approaches described earlier in this section. Table 5 presents the results, with the best approach for each data source in bold.

Table 6 Precision measures (%) for each algorithm in each source

Algorithm	CE	NYT	Yahoo	Wiki	BBC	Ars	Chaos	Ave.
BT	88.87	62.22	54.94	83.91	69.09	68.25	76.36	71.95
CCB	80.61	57.61	46.90	63.22	53.52	64.05	64.45	61.48
DSC	91.94	98.58	96.54	81.67	89.27	95.82	94.45	92.61
FE	73.87	97.51	**99.08**	**98.79**	**98.95**	0.005	72.59	77.26
KFE	79.28	73.82	69.49	73.76	63.84	81.35	73.97	73.64
LQF	88.60	90.02	64.54	83.60	77.03	88.40	82.76	82.14
CETR	91.26	85.19	69.36	94.69	68.93	83.06	78.75	81.61
CETD-DS	92.96	98.38	83.16	98.31	84.39	97.81	93.59	92.66
CECTD-S	90.35	96.72	80.33	98.02	82.55	94.61	89.64	90.33
CECTD-DS	**95.87**	99.69	84.59	98.25	86.15	98.04	**96.21**	94.11
CEHTD-DS	94.97	**99.72**	91.99	96.58	95.55	**98.12**	94.74	**95.95**

The best scores are marked in bold

Table 7 Recall measures (%) for each algorithm in each source

Algorithm	CE	NYT	Yahoo	Wiki	BBC	Ars	Chaos	Ave.
BTE	95.83	98.38	**95.06**	81.60	97.09	97.91	92.80	94.10
CCB	92.71	96.09	93.45	73.14	92.19	96.27	91.05	90.70
DSC	62.01	85.67	73.14	34.61	78.89	90.52	80.27	72.16
FE	9.97	3.62	4.94	1.48	3.71	0.001	6.22	4.28
KFE	69.61	71.35	56.97	44.60	65.02	82.73	66.04	65.19
LQF	94.02	97.10	90.65	76.41	92.17	98.43	93.98	91.82
CETR	86.08	90.58	77.65	72.77	86.58	93.93	86.92	84.93
CETD-DS	94.52	95.84	85.90	97.22	95.21	98.85	94.99	94.65
CECTD-S	92.60	96.56	93.34	**97.61**	93.77	93.56	91.22	93.97
CECTD-DS	**97.15**	**98.16**	93.99	92.77	**97.95**	**99.51**	96.10	**96.52**
CEHTD-DS	94.07	95.96	88.59	90.41	96.46	98.83	**96.42**	94.39

The best scores are marked in bold

Interestingly, the DSC algorithm does not perform better than the BTE algorithm, even though it actually extends the BTE algorithm. We believe this is due to the windowing technique which improves the precision, but reduces the recall rate of the DSC algorithm.

It is noteworthy that CETR is very similar to our approach, whereby content is extracted by tag ratios. However, CETR loses the structural information of web pages, since the tag ratios are computed on a line-by-line basis. And the results in Tables 5, 6 and 7 show that our CETD methods outperform CETR. On the other hand, although VIPS is a visual-information-based method, which cannot be compared directly as it outputs a set of page segments rather than extracted text, it can be deduced that CETD performs better than VIPS since CETR outperforms VIPS in [33].

Tables 6 and 7 show that other methods typically achieve either a high recall or a high precision but rarely both. Our methods performs more consistently than other algorithms with overall high scores in both metrics, as the average F_1 scores of CECTD-DS and CEHTD-DS are 8.7 % higher than the best score among other approaches.

4.3 Discussion

The results show that CETD is an effective and robust content extraction algorithm, which performs relatively well even on non-news web corpora with considerable diversity, such as the CleanEval data set. Note that CETD does not require manual-labeled training examples since it is a completely unsupervised algorithm.

The results also show that CECTD-DS outperforms CETD-DS and CECTD-S on broader corpora. It shows the Composite Text Density and DensitySum techniques are more effective than Text Density and Data Smoothing. Furthermore, with the visual importance information incorporated, CEHTD-DS also performs better than CETD-DS and CECTD-S and is more suitable than CECTD-DS when extraction results are expected to be more precise.

In contrast to most other methods, which just output unformatted text, content with complete structure information can be obtained by CETD because all operations are done in the DOM tree. Therefore, CETD can be easily integrated with other applications.

Many current methods only extract the most probable content section. However, there is no rule that a web page may only have a single content section. There are many web pages, especially blogs where content is separated by horizontal lines or other delimiters. CETD is not affected by multiple content sections.

Despite many advantages of our algorithm, there are some weaknesses. It does not perform well with some site categories, such as video and picture sites. The contents of such sites are videos or pictures as well as the comments under them. Therefore, our findings do not hold for these sites. For YouTube, CETD can only extract comments. Another situation where CETD does not perform well is portal home pages. These pages usually contain a vast array of menus and news titles or short news descriptions, and most of these are hyperlinks. CETD has problems discerning the content section(s) of these types of web pages. However, in general, these pages have no topic. Therefore, extracting the contents of these pages does not make much sense.

5 Conclusions

In this paper, we have proposed a method for extracting the contents from web pages by Text Density, based on the observation that the content text is usually lengthy and simply formatted, while noise is usually highly formatted and contains less text with brief sentences. Observing that noise contains many more hyperlinks than meaningful content, we extended Text Density to Composite Text Density by adding statistical information about hyperlinks. Furthermore, the visual information of relative displaying positions and sizes is metered by Visual Importance and incorporated into the Hybrid Text Density. In order to extract the content completely, we proposed the DensitySum technique instead of Data Smoothing. The effectiveness of the CETD algorithm has been demonstrated. The results show that CETD performs better than several other content extraction algorithms.

In addition to its effectiveness, the greatest strength of this algorithm over other methods is the simplicity of its concept and implementation. Furthermore, this algorithm just requires a web page as input, and then returns a cleaned page without adjusting parameters, training and building classifier models. CETD can also retain the original structure information of the web pages, which can then be utilized for other applications, such as small-screen devices.

6 Future work

An intuitive and simple computation of the visual information is conducted in this paper, which only considers the relative displaying positions and sizes of leaf DOM nodes. In contrast, there are many other features with abundant visual information that can be added to our approach in the future. One option is the page segmentation results attained by the VIPS [5]. As mentioned before, the VIPS is a visual-information-based method, which outputs a set of page segments rather than extracted texts. It could be a good resource for visual features extraction, which can be adopted into future improvements of our method. Additionally, as the implemented CECTD-DS and CEHTD-DS methods have their respective characteristics for different Web sites, how to automatically leverage their advantages will be further explored. Additionally, as considered in [11], some print links and images are also important resources, which will be further considered by extending our approach.

Acknowledgments This work is funded by the National Program on Key Basic Research Project (973 Program, Grant No. 2013CB329605), Natural Science Foundation of China (NSFC, Grant Nos. 60873237 and 61003168), Natural Science Foundation of Beijing (Grant No. 4092037), Outstanding Young Teacher Foundation and Basic Research Foundation of Beijing Institute of Technology, and partially supported by Beijing Key Discipline Program.

References

1. Adelberg B (1998) NoDoSE—a tool for semi-automatically extracting semi-structured data from text documents. In: Proceedings of SIGMOD '98. ACM, New York, NY, USA, pp 283–294
2. Baluja S (2006) Browsing on small screens: recasting web-page segmentation into an efficient machine learning framework. In: Proceedings of WWW '06, pp 33–42
3. Bar-Yossef Z, Rajagopalan S (2002) Template detection via data mining and its applications. In: Proceedings of WWW '02. NY, USA, New York, pp 580–591
4. Bu Z, Zhang C, Xia Z, Wang J (2013) An FAR-SW based approach for webpage information extraction. Inf Syst Front 1–15. doi:10.1007/s10796-013-9412-2
5. Cai D, Yu S, Wen J, Ma W (2003) Extracting content structure for web pages based on visual representation. In: Proceedings of APWeb'03, pp 406–417
6. Chen L, Ye S, Li X (2006) Template detection for large scale search engines. In: Proceedings of SAC '06. NY, USA, New York, pp 1094–1098
7. Chen Y, Fankhauser P, Zhang H-J (2003) Detecting web page structure for adaptive viewing on small form factor devices. In: Proceedings of WWW '03, pp 225–233
8. Davison BD (2000) Recognizing nepotistic links on the web. In: AAAI-2000 workshop on artificial intelligence for web search. Austin, TX, pp 23–28
9. Debnath S, Mitra P, Giles CL (2005) Automatic extraction of informative blocks from webpages. In: Proceedings of SAC '05, pp 1722–1726
10. Debnath S, Mitra P, Giles CL (2005) Identifying content blocks from web documents. ISMIS 3488(5):285–293
11. Fan J, Luo P, Lim SH, Liu S, Parag J, Liu J (2011) Article clipper: a system for web article extraction. In: Proceedings of KDD '11, pp 743–746
12. Fernandes D, de Moura ES, Ribeiro-Neto B, da Silva AS, Gonçalves MA (2007) Computing block importance for searching on web sites. In: Proceedings of CIKM '07, pp 165–174
13. Finn A, Kushmerick N, Smyth B (2001) Fact or fiction: content classification for digital libraries. In: Joint DELOS-NSF workshop: personalization and recommender systems in digital libraries
14. Fumarola F, Weninger T, Barber R, Malerba D, Han J (2011) Extracting general lists from web documents: a hybrid approach. In: Proceedings of IEA/AIE '11. Heidelberg, Berlin, pp 285–294
15. Gibson D, Punera K, Tomkins A (2005) The volume and evolution of web page templates. In: Proceedings of WWW '05. ACM, New York, NY, USA, pp 830–839
16. Gottron T (2008) Combining content extraction heuristics: the CombinE system. In: Proceedings of iiWAS '08, pp 591–595

17. Gottron T (2008) Content code blurring: a new approach to content extraction. In: Proceedings of DEXA '08, pp 29–33
18. Gupta S, Kaiser G, Stolfo S (2005) Extracting context to improve accuracy for HTML content extraction. In: Proceedings of WWW '05, pp 1114–1115
19. Kao H, Lin S, Ho J, Chen M (2004) Mining web informative structures and contents based on entropy analysis. IEEE Trans Knowl Data Eng 16:41–55
20. Kohlschütter C, Fankhauser P, Nejdl W (2010) Boilerplate detection using shallow text features. In: Proceedings of WSDM '10, pp 441–450
21. Kushmerick N (1999) Learning to remove internet advertisements. In: Proceedings of AGENTS '99. NY, USA, New York, pp 175–181
22. Li Y, Dong S-b, Zheng X, Ma B-H (2012) Improving navigation page detection by using DOM-based block text identification. In: Proceedings of 10th international conference on ICT and knowledge engineering, Bangkok, pp 129–134
23. Lin S, Ho J (2002) Discovering informative content blocks from web documents. In: Proceedings of SIGKDD '02. NY, USA, New York, pp 588–593
24. Liu L, Pu C, Han W (2000) XWRAP: an XML-enabled wrapper construction system for web information sources. In: Proceedings of ICDE '00, pp 611–621
25. Mantratzis C, Orgun M, Cassidy S (2005) Separating XHTML content from navigation clutter using DOM-structure block analysis. In: Proceedings of HYPERTEXT '05, pp 145–147
26. Marek M, Pecina P, Spousta M (2007) Web page cleaning with conditional random fields. In: Proceedings of WAC3 '07, Cleaneval session
27. Peters ME, Lecocq D (2013) Content extraction using diverse feature sets. In: Proceedings of WWW '13. Republic and Canton of Geneva, Switzerland, pp 89–90
28. Pinto D, Branstein M, Coleman R, Croft WB, King M, Li W, Wei X (2002) QuASM: a system for question answering using semi-structured data. In: Proceedings of JCDL '02, pp 46–55
29. Rahman AFR, Alam H, Hartono R (2001) Content extraction from HTML documents. In: Proceedings of WDA '01, pp 7–10
30. Shen D, Wang H, Jiang Z, Cao J (2013) A high efficient incremental microblog crawler: design and implementation. J Inf Comput Sci 10(6):1731–1747
31. Song R, Liu H, Wen J, Ma W (2004) Learning block importance models for web pages. In: Proceedings of WWW '04. NY, USA, New York, pp 203–211
32. W3C Document Object Model (2009) Website. http://www.w3.org/DOM
33. Weninger T, Hsu WH, Han J (2010) CETR—content extraction via tag ratios. In: Proceedings of WWW '10. NY, USA, New York, pp 971–980
34. Yi L, Liu B, Li X (2003) Eliminating noisy information in web pages for data mining. In: Proceedings of SIGKDD '03. NY, USA, New York, pp 296–305

Author Biographies

Dandan Song received a B.E. degree and a Ph.D. degree from the Department of Computer Science and Technology, Tsinghua University, Beijing, China, in 2004 and 2009, respectively. She is currently an assistant professor in the School of Computer Science and Technology, Beijing Institute of Technology. Her research interests include information retrieval, data mining and bioinformatics.

Fei Sun received a B.E. degree and a Master degree from the School of Computer Science and Technology, Beijing Institute of Technology, Beijing, China, in 2009 and 2012, respectively. He is currently working toward the Ph.D. degree from the Institute of Computing Technology, Chinese Academy of Sciences. His research interests include information retrieval and web data mining.

Lejian Liao received a Ph.D. degree from the Institute of Computing Technology, Chinese Academy of Sciences. He is currently a Professor in the School of Computer Science and Technology, Beijing Institute of Technology, Beijing, China. He also served as Vice Dean of the school. With main research interest in machine learning, natural language processing and intelligent network, Professor Lejian Liao has published numerous papers in several areas of computer science.