DOM Based Content Extraction via Text Density

Fei Sun, Dandan Song* and Lejian Liao*
Lab of High Volume language Information Processing & Cloud Computing
Beijing Lab of Intelligent Information Technology
School of Computer Science, Beijing Institute of Technology
Beijing, China
{ofey, sdd, liaolj}@bit.edu.cn

ABSTRACT

In addition to the main content, most web pages also contain navigation panels, advertisements and copyright and disclaimer notices. This additional content, which is also known as noise, is typically not related to the main subject and may hamper the performance of web data mining, and hence needs to be removed properly. In this paper, we present Content Extraction via Text Density (CETD)—a fast, accurate and general method for extracting content from diverse web pages, and using DOM (Document Object Model) node text density to preserve the original structure. For this purpose, we introduce two concepts to measure the importance of nodes: Text Density and Composite Text Density. In order to extract content intact, we propose a technique called DensitySum to replace Data Smoothing. The approach was evaluated with the CleanEval benchmark and with randomly selected pages from well-known websites, where various web domains and styles are tested. The average F_1-scores with our method were 8.79% higher than the best scores among several alternative methods.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Information filtering; H.3.1 [**Content Analysis and Indexing**]: Abstracting methods

General Terms

Algorithms, Experimentation

Keywords

Content Extraction, Text Density, Composite Text Density, DensitySum

1. INTRODUCTION

The explosive growth of the Internet has produced a huge number of information sources, and its influence continues to increase. Thus, web data mining has become an important and popular technique for discovering useful information or knowledge. Such research requires *main content* (*e.g.*, an article text) from the web to be gathered, processed and stored quickly and efficiently. However, the 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 website owners and benefits user browsing, it is unrelated to the topic of the web pages and not simple enough for computer programs to parse. Thus this information can be treated as noise, hampering information gathering and web mining.

In order to improve the performance of web mining and information retrieval from web pages, content extraction techniques have been developed to remove such noise. Generally, content extraction improves performance, and is essential for many real world applications.

Traditionally, building text corpora was 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 [23].

Additionally, as Pocket-sized devices with small screens such as PDAs and mobile phones have become ubiquitous, adapting web pages for small screens has become an increasingly important and challenging task [3, 7].

However, extracting the main content from the web pages has become more difficult and nontrivial. As early as 2005, Gibson et al. [13] 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. Meanwhile, web page layout has become much more complex than before, especially with the development and widespread use of the Cascading Style Sheets (CSS) technique. We find that most of the recent web pages use the style sheets and <div> or <span> tags for structural information to replace structural tags within a web page. Many early content extraction techniques failed to keep up with these changes and performed poorly, because recent web pages no longer include particular HTML cues (*e.g.*, <table>, <td> and fonts) that they used before.

In addition, nowadays most additional web page content, especially advertisements (*e.g.*, Google AdSense), are gen-

*Corresponding authors.

erated dynamically, which means template detection algorithms perform poorly. Moreover, they may also break easily due to the changes in page structure.

In this paper, we propose a highly effective content extraction algorithm to extract the main content from web pages. It can not only extract the main content, but also preserve its original structure information.

Our content extraction technique is based on the following observation: 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. 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.

First, we propose two measures for the importance of the tags in the 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. Furthermore, we extend the Text Density to the Composite Text Density by adding statistical information about hyperlinks. 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 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. They are not very 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 section 2, we propose two different definitions for text density in section 3. Next, we describe 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. Finally we offer our conclusions and plans for future research.

2. RELATED WORK

The term *Content Extraction* (CE) was introduced by Rahman et al. in 2001 [25]. 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 [2] or XWRAP [21]) extracted article text embedded in a common template from web pages by looking for some HTML cues, using regular expressions. These 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 website. Still, even individual websites employ multiple structures; furthermore, these websites may also change structures or layouts over time. All the above situations mean such approaches require constant updating.

Kushmerick [19] and Davison [8] proposed machine learning mechanisms to recognize banner advertisements, redundant and irrelevant links in 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.

The Vision-based Page Segmentation (VIPS) technique was introduced by Cai et al. [5] in 2003 to divide a web page into a tree, where the nodes are visually grouped blocks. Based on the VIPS method, Song et al. [26] presented an approach to rank block importance for web pages through learning algorithms using spatial features (such as position or size) and content features (such as the number of images and links); and Fernandes et al. [11] developed another way to compute block importance of a web page by means of assigning weights to classes of structurally similar blocks. However, VIPS must partially render a page in order to analyze it. Additionally, if external style sheets are used, they should also be retrieved. Therefore, compared to other techniques, VIPS is resource intensive.

A different approach for content extraction is Template Detection (TD) algorithms [4, 20, 28, 6, 17] in which collections of documents based on the same template are used to learn a common structure. Bar-Yossef et al. presented an approach to automatically detect templates from the largest *pagelet*, *i.e.* self-contained regions in a web page [4]. Lin et al. partitioned a page with <table> tags, and identified redundant blocks using an entropy measure over a set of word-based features [20]. In order to improve the performance of web page clustering and classification, Yi et al. introduced a *site style tree* (SST) structure that labels DOM nodes with similar styles across pages as uninformative [28]. 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 website. 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 [23].

There are sets of content extraction approaches based on statistical information of web pages. In 2001, Finn et al. introduced the Body Text Extraction (BTE) algorithm to improve the accuracy of the content's classifier for digital libraries [12]. 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. [24] 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. 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 [22]. 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. proposed the FeatureExtractor (FE) and KFeatureExtractor (KFE) techniques based on block segmentation of the HTML document [9, 10]. 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 in 2008 [15]. Content regions are detected in homogeneously formatted source code character sequences.

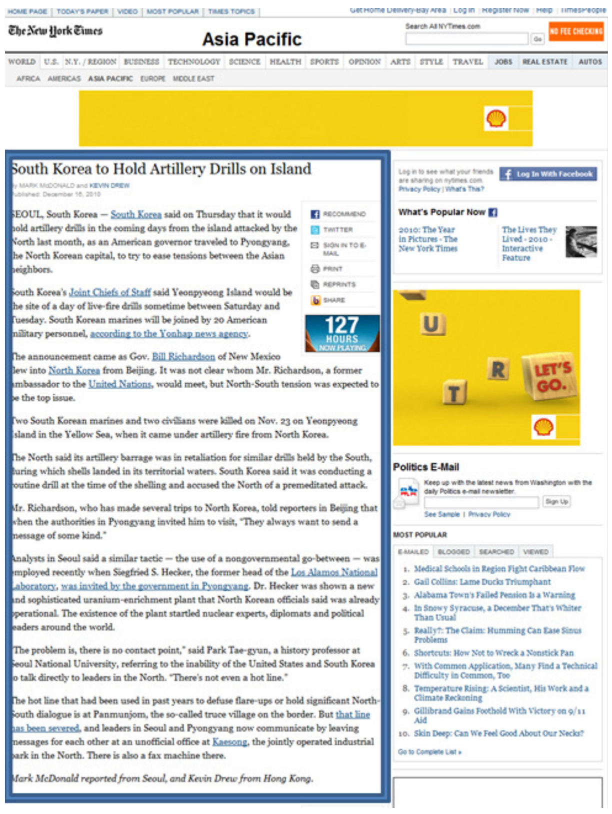
Weninger et al. introduced the Content Extraction via Tag Ratios (CETR) algorithm, a method to extract content text from diverse web pages using the HTML document's tag ratios [27]. The approach computes tag ratios on a line-by-line basis and then clusters the resulting histogram into content and noise areas. This is a laconic and efficient algorithm, however vulnerable to the page's source code style changes.

Kohlschütter et al. developed a simple, yet effective technique to classify individual text elements from a web page [18]. 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. attempted to combine different heuristics into one system called the *Crunch* framework [16]. 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 developed *CombineE* framework, a more recent ensemble method, which made it easier to configure ensembles of content extraction algorithms [14].

3. TEXT DENSITY

Let's take the news article from The New York Times[1] shown in Figure 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.


HOME PAGE | TODAY'S PAPER | VIDEO | MOST POPULAR | TIMES TOPICS
Get Home Delivery-Bay Area | Log In | Register Now | Help | TimesPeople
The New York Times
Asia Pacific
Search All NYTimes.com
Go
NO FEE CHECKING
WORLD U.S. N.Y. / REGION BUSINESS TECHNOLOGY SCIENCE HEALTH SPORTS OPINION ARTS STYLE TRAVEL JOBS REAL ESTATE AUTOS
AFRICA AMERICAS ASIA PACIFIC EUROPE MIDDLE EAST
RECOMMEND
TWITTER
SIGN IN TO E-MAIL
PRINT
REPRINTS
SHARE
127 HOURS NOW PLAYING
Log In With Facebook
U
R
T
LET'S GO.
Sign Up
E-MAILED
BLOGGED
SEARCHED
VIEWED

outh Korea to Hold Artillery Drills on Island

y MARK McDONALD and KEVIN DREW
ublished: December 16, 2010

EOUL, South Korea — South Korea said on Thursday that it would old artillery drills in the coming days from the island attacked by the orth last month, as an American governor traveled to Pyongyang, he North Korean capital, to try to ease tensions between the Asian eighbors.

outh Korea's Joint Chiefs of Staff said Yeonpyeong Island would be he site of a day of live-fire drills sometime between Saturday and uesday. South Korean marines will be joined by 20 American military personnel, according to the Yonhap news agency.

he announcement came as Gov. Bill Richardson of New Mexico lew into North Korea from Beijing. It was not clear whom Mr. Richardson, a former mbassador to the United Nations, would meet, but North-South tension was expected to e the top issue.

wo South Korean marines and two civilians were killed on Nov. 23 on Yeonpyeong sland in the Yellow Sea, when it came under artillery fire from North Korea.

he North said its artillery barrage was in retaliation for similar drills held by the South, uring which shells landed in its territorial waters. South Korea said it was conducting a outine drill at the time of the shelling and accused the North of a premeditated attack.

Ir. Richardson, who has made several trips to North Korea, told reporters in Beijing that vhen the authorities in Pyongyang invited him to visit, "They always want to send a nessage of some kind."

nalysts in Seoul said a similar tactic — the use of a nongovernmental go-between — was mployed recently when Siegfried S. Hecker, the former head of the Los Alamos National aboratory, was invited by the government in Pyongyang. Dr. Hecker was shown a new nd sophisticated uranium-enrichment plant that North Korean officials said was already perational. The existence of the plant startled nuclear experts, diplomats and political eaders around the world.

The problem is, there is no contact point," said Park Tae-gyun, a history professor at eoul National University, referring to the inability of the United States and South Korea o talk directly to leaders in the North. "There's not even a hot line."

he hot line that had been used in past years to defuse flare-ups or hold significant North-outh dialogue is at Panmunjom, the so-called truce village on the border. But that line as been severed, and leaders in Seoul and Pyongyang now communicate by leaving nessages for each other at an unofficial office at Kaesong, the jointly operated industrial ark in the North. There is also a fax machine there.

Iark McDonald reported from Seoul, and Kevin Drew from Hong Kong.

Log in to see what your friends are sharing on nytimes.com.
Privacy Policy | What's This?

What's Popular Now

2010: The Year in Pictures - The New York Times

The Lives They Lived - 2010 - Interactive Feature

Politics E-Mail

Keep up with the latest news from Washington with the daily Politics e-mail newsletter.

See Sample | Privacy Policy

MOST POPULAR

1. Medical Schools in Region Fight Caribbean Flow
2. Gail Collins: Lame Ducks Triumphant
3. Alabama Town's Failed Pension Is a Warning
4. In Snowy Syracuse, a December That's Whiter Than Usual
5. Really?: The Claim: Humming Can Ease Sinus Problems
6. Shortcuts: How Not to Wreck a Nonstick Pan
7. With Common Application, Many Find a Technical Difficulty in Common, Too
8. Temperature Rising: A Scientist, His Work and a Climate Reckoning
9. Gillibrand Gains Foothold With Victory on 9/11 Aid
10. Skin Deep: Can We Feel Good About Our Necks?

Go to Complete List »

Figure 1: The New York Times web page article

To extract content from this web page, we take advantage of typical text 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 Figure 1 also supports this observation.

3.1 DOM Tree

Document Object Model (DOM) [1] 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.

Example 1. *Below is a brief segment of HTML code from Figure 1.*

```
<div class="main">
  <div class="article">
    <div class="articleHeadline">
      South Korea to Hold Artillery Drills on Island</div>
    <div class="articleBody">
    ...The announcement came as...
    <a>Bill Richardson</a>...
</div></div></div>
```

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

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

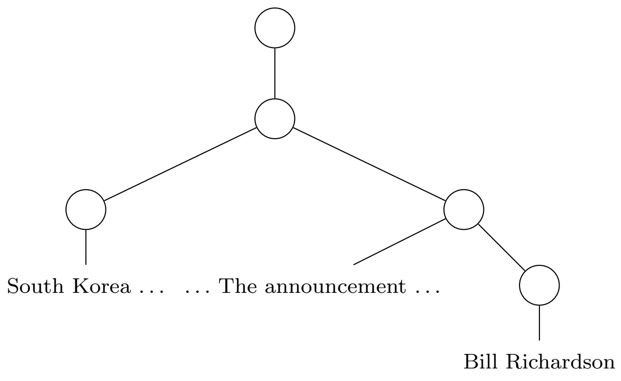


Figure 2: The DOM tree of Example 1.

Notice that 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

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 the ratios 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 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:

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

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, it should be set to 1.

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. In the likely case where the number of tags under a particular node is 0, the density is set to the number of characters the node contains. 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 2 below shows the text density for each node of Example 1.

Example 2. *The text density for the five tags in Example 1 are computed as follows:*

Algorithm 1 Pseudocode of ComputeDensity(N)

```
 1: INPUT: $N$
 2: OUTPUT: $N$
 3: for all child node $C$ in $N$ do
 4:     $ComputeDensity(C)$
 5: end for
 6: $N.CharNumber \leftarrow CountChar(N)$
 7: $N.TagNumber \leftarrow CountTag(N)$
 8: if $N.TagNumber == 0$ then
 9:     $N.TagNumber \leftarrow 1$
10: end if
11: $N.Density \leftarrow N.CharNumber/N.TagNumber$
```

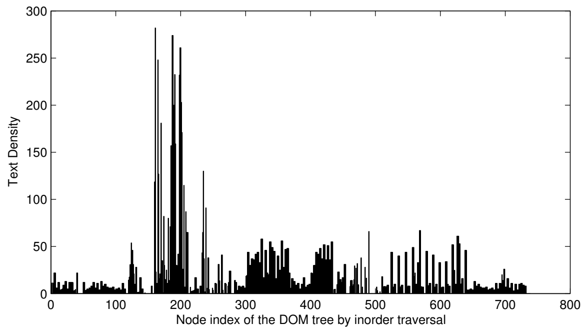


Figure 3: Text density for each node from NYtimes web page

1. `<div "main">`: `Chars=85`, `Tags=4`, `Density=21.25`
2. `<div "article">`: `Chars=85`, `Tags=3`, `Density=28.33`
3. `<div "articleHeadline">`: `Chars=46`, `Tags=1`, `Density=46`
4. `<div "articleBody">`: `Chars=39`, `Tags=1`, `Density=39`
5. `<a>`: `Chars=15`, `Tags=1`, `Density=15`

Figure 3 shows the resulting density-histogram for the page in Figure 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.

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 Figure 1 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*. Unless otherwise specified, we use Text Density to refer to these two densities.

Definition 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:

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

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 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 2, $\frac{C_i}{LC_i}$ is a measure of the proportion of hyperlink texts in i; accordingly, $\frac{T_i}{LT_i}$ is a measure 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 an extremely high values, or nodes which contain brief text (*e.g.*, news headlines or one sentence paragraphs) from getting extremely low values.

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 Figure 3, Composite Text Density, shown in Figure 4, is better suited for classification because of the more obvious differences between sections. These two calculation methods of Text Density are discussed further in Section 5.

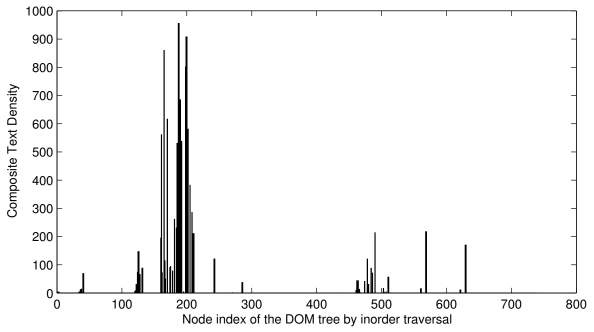


Figure 4: Composite text density for each node from NYtimes web page

4. CONTENT EXTRACTION

In this section 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. Simply, any node with text density greater than or equal to t should be labeled as content; and any node should be labeled as noise if its text density is less than t. The problem then becomes a matter of finding the best value of t and a way to extracting the content completely.

4.1 Threshold

Intuitively, since the `<body>` tag is the root node that we compute, therefore it should contains both content and noise. This means that it contains more text than noise, and more hyperlinks than meaningful content. Hence, its text density should be an intermediate value sufficient to distinguish between the two. Empirically, we set the `<body>` tag's text density as the threshold in our study. It must be noted that this criterion is slightly modified in section 4.2.

4.2 DensitySum

It is easy to see that some nodes' text density is abnormally different from the surrounding nodes in Figure 3 and 4. 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 information) may get an abnormally high text density. The authors of [18] also noted this fact. 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.

To solve this problem, we propose a technique called *DensitySum.* 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 noisy nodes may be retained because of text density increase.

DensitySum, 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. 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:

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

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 or Composite Text Density in practice.

For a simple case, if there exists just one content block, we identify the content by looking for the node under the `<body>` tag with the maximum DensitySum, then mark it as content. Afterwards, we extract the content by keeping its ancestors and subtrees.

Note that in many cases, the web page contains more than one content block. Therefore, for each node in which the text density is greater than the threshold, we apply the same method described above to extract content.

Moreover, in some web pages, the content block's text density may be less than the `<body>`'s text density. This block will be lost if we simply use the `<body>`'s text density as threshold. 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. 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:     $T \leftarrow FindMaxDensitySumTag(N)$
4:     $MarkContent(T)$
5:     for all child node $C$ in $N$ do
6:         $ExtractContent(C)$
7:     end for
8: end if
```

4.3 Implementation Details

The program is implemented using Tidy[2] and RapidXml[3] in C++. Since there are several types of errors (*e.g.*, non-standard tags, incorrect encoding of HTML files) in web pages, parsing these web pages directly would fail. We use Tidy to correct such errors, and output them as standard XHTML documents. Afterwards, the XHTML documents are parsed into DOM trees using RapidXml. 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 webpage, buttons and drop-down lists are treated as hyperlinks.

Three distinct algorithms are implemented. The first is the Text Density with DensitySum method, hereafter referred to as CETD-DS. The second is the Composite Text Density with Data Smoothing method, which, hereafter referred to as CECTD-S; here it is simply weighted averaging with sibling nodes. The last one is the Composite Text Density with DensitySum method, hereafter referred to as CECTD-DS. We use Content Extraction via Text Density (CETD) to refer to the three algorithms collectively.

5. EXPERIMENT

In this section we describe experiments on real world data from various web sites to demonstrate the effectiveness of our method. Data sets used in our experiments and evaluation measures are described first. We then present and discuss our experimental results.

5.1 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 [23]. 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 use various styles and structures.

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. 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.

CETD: In order to evaluate our methods on varied sources, we produced a data set (which can be obtained from the author's website[4]). This data set is separated into two non-overlapping sets. (1) The *Big 5*: Ars Technica, BBC, Yahoo!, New York Times, Wikipedia, 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.

5.2 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:

$$P = \frac{LCS(a,b).length}{a.length}, R = \frac{LCS(a,b).length}{b.length} \quad (1)$$

$$F_1 = \frac{2 \times P \times R}{P + R} \quad (2)$$

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 further 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:

$$Score(a,b) = 1 - \frac{distance(a,b)}{alignmentLength(a,b)} \quad (3)$$

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

[2]`http://tidy.sourceforge.net/`
[3]`http://rapidxml.sourceforge.net/`

[4]`http://ofey.me/projects/cetd/`

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%

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%

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:

$$Score(a, b) = \frac{LCS(a, b).length}{a.length + b.length - LCS(a, b).length} \quad (4)$$

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

5.3 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 Section 2 have been implemented in Java for the Combine*E* framework [14]. In addition, CETR was also implemented in Java [27]. Evaluation was performed by providing each HTML document as input to each algorithm and collecting the results.

5.4 Result

All results are collected by calculating the average of each metrics over all examples.

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.

Table 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.

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 tags to a low density value, even lower than the threshold. However, the Text Density method could be unaffected.

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%

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

Clearly, Table 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 for 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.

Overall, these results show that the CECTD-DS performs far better than CETD-DS and CECTD-S. It is exciting that the CleanEval scores are higher than the winner of CleanEval competition, which only scored 84.1% on the English data set [23].

The results show that the precision of all 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 computed because they are not visible in the browser, are included in the results.

5.4.1 Methods Comparison

In order to show the effectiveness of our methods, we compare the above performance with the alternative approaches described earlier in this section. Table 4 presents the results with the best approach for each data source 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. The results in Table 4, 5 and 6 show that the Composite Text Density and DensitySum methods outperform CETR.

Table 5 and 6 show that other methods typically achieve either a high recall or a high precision but rarely both. Our

CETD performs more consistently than other algorithms with overall high scores in both metrics, and CECTD-DS is the best performing algorithm in most data sets with the average F_1-score 8.79% higher than the best score among other approaches..

5.5 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.

For practical purposes, users usually value recall over precision as a performance metric. Although, CECTD-DS can achieve a very high recall rate, users can still see a marked increase in recall and a sharp decrease in precision by reducing the threshold. This precision/recall tradeoff is shown in Figure 5. We set the threshold in actual use as the base value, and use a coefficient λ to adjust the threshold value in this experiment. Note that the threshold must be lower than the text density of the `<body>` tag, otherwise the whole page will be viewed as noise. When $\lambda = 0$, the recall is always 100% because all nodes in DOM are included. For the Yahoo! domain, shown in Figure 5, a good tradeoff might be $\lambda = 0.9$. Finding a good threshold value is difficult. In current experiments, we set $\lambda = 1$, which can already achieve very good results on various domains.

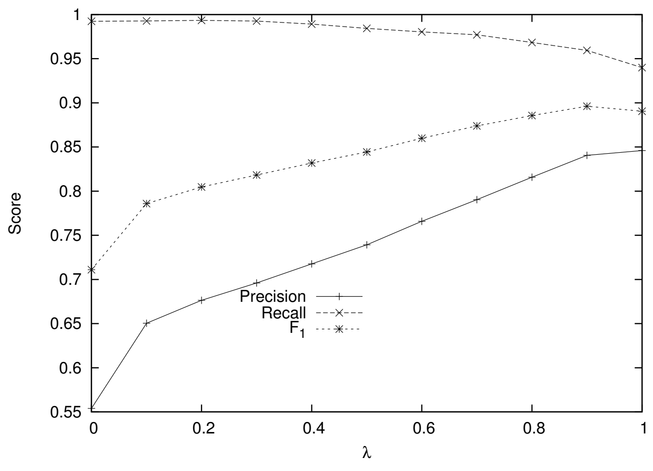


Figure 5: Precision, Recall and F_1 tradeoff for Yahoo! as the threshold coefficient (λ) increases

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.

6. 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 contain 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. 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.

6.1 Future Work

In the research for this paper, we found that HTML Tidy was not sufficiently robust since it may sometimes cause some pages to not be properly parsed. In the future, the WebKit[5] Kernel will be incorporated to parse web pages, so that pages rendered normally by a browser can be parsed correctly.

Another area for further investigation is to identify the hidden elements of the pages. In some sites, some of the non-content part (*e.g.*, BBC disclaimers) is not visible. We may achieve better results if we can remove these non-visible elements before the computation of text density, since these elements are meaningless for the end user but treated as non-tag text in the algorithm.

Intuitively, the meaningful content always occupies the center of the screen. Obviously, a web page can be partitioned into multiple segments or blocks, and usually the importance of those blocks on a page is not equivalent. Therefore, spatial information (*e.g.*, position, size) will be added to the density measure to further enhance the performance.

Finally, in this paper, we focused on the comparison between the content extraction methods based on statistical

[5]`http://webkit.org/`

Table 4: F_1-measures for each algorithm in each source. The best scores are marked in bold.

Algorithm	CleanEval-Eng	NYTimes	Yahoo!	Wikipedia	BBC	Ars Technica	Chaos	Average
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%**

Table 5: Precision-measures for each algorithm in each source. The best scores are marked in bold.

Algorithm	CleanEval-Eng	NYTimes	Yahoo!	Wikipedia	BBC	Ars Technica	Chaos	Average
BTE	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%**

Table 6: Recall-measures for each algorithm in each source. The best scores are marked in bold.

Algorithm	CleanEval-Eng	NYTimes	Yahoo!	Wikipedia	BBC	Ars Technica	Chaos	Average
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%**

information of web pages. VIPS, on the other hand, 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 outperform VIPS in [27]. In a future study, we will do more experiments to verify this.

7. ACKNOWLEDGMENTS

The authors sincerely thank Tim Weninger, the author of CETR, for the fruitful discussions and data access. We also thank Thomas Gottron, author of the *CombineE* framework, for some of the implementations used in this work; and the CleanEval team for providing a standard evaluation data set. This work is funded by NSFC (Grant Nos.60873237 and 61003168), Beijing Natural Science Foundation (Grant No.4092037), Excellent Researcher Award Program and Basic Research Foundation of Beijing Institute of Technology.

8. REFERENCES

[1] W3C document object model. Website, 2009. http://www.w3.org/DOM.

[2] B. Adelberg. Nodose—a tool for semi-automatically extracting semi-structured data from text documents. In *Proceedings of SIGMOD '98*, pages 283–294, New York, NY, USA, 1998. ACM.

[3] S. Baluja. Browsing on small screens: recasting web-page segmentation into an efficient machine learning framework. In *Proceedings of WWW '06*, pages 33–42, 2006.

[4] Z. Bar-Yossef and S. Rajagopalan. Template detection via data mining and its applications. In *Proceedings of WWW '02*, pages 580–591, New York, NY, USA, 2002.

[5] D. Cai, S. Yu, J. Wen, and W. Ma. Extracting content structure for web pages based on visual representation. In *Proceedings of APWeb'03*, pages 406–417, 2003.

[6] L. Chen, S. Ye, and X. Li. Template detection for large scale search engines. In *Proceedings of SAC '06*, pages 1094–1098, New York, NY, USA, 2006.

[7] Y. Chen, P. Fankhauser, and H.-J. Zhang. Detecting web page structure for adaptive viewing on small form factor devices. In *Proceedings of WWW '03*, pages 225–233, 2003.

[8] B. D. Davison. Recognizing nepotistic links on the web. In *AAAI-2000 Workshop On Artificial Intelligence For Web Search*, pages 23–28, Austin, Texas, 2000.

[9] S. Debnath, P. Mitra, and C. L. Giles. Automatic extraction of informative blocks from webpages. In *Proceedings of SAC '05*, pages 1722–1726, 2005.

[10] S. Debnath, P. Mitra, and C. L. Giles. Identifying content blocks from web documents. *ISMIS*, 3488(5):285–293, November 2005.

[11] D. Fernandes, E. S. de Moura, B. Ribeiro-Neto, A. S. da Silva, and M. A. Gonçalves. Computing block importance for searching on web sites. In *Proceedings of CIKM '07*, pages 165–174, 2007.

[12] A. Finn, N. Kushmerick, and B. Smyth. Fact or fiction: Content classification for digital libraries. In *Joint DELOS-NSF Workshop: Personalization and Recommender Systems in Digital Libraries*, 2001.

[13] D. Gibson, K. Punera, and A. Tomkins. The volume and evolution of web page templates. In *WWW '05*, pages 830–839, New York, NY, USA, 2005. ACM.

[14] T. Gottron. Combining content extraction heuristics: the *CombinE* system. In *Proceedings of iiWAS '08*, pages 591–595, 2008.

[15] T. Gottron. Content code blurring: A new approach to content extraction. In *Proceedings of DEXA '08*, pages 29–33, 2008.

[16] S. Gupta, G. Kaiser, and S. Stolfo. Extracting context to improve accuracy for html content extraction. In *Proceedings of WWW '05*, pages 1114–1115, 2005.

[17] H. Kao, S. Lin, J. Ho, and M. Chen. Mining web informative structures and contents based on entropy analysis. In *IEEE Transactions on Knowledge and Data Engineering*, pages 41–55, Piscataway, NJ, USA, 2004.

[18] C. Kohlschütter, P. Fankhauser, and W. Nejdl. Boilerplate detection using shallow text features. In *Proceedings of WSDM '10*, pages 441–450, 2010.

[19] N. Kushmerick. Learning to remove internet advertisements. In *Proceedings of AGENTS '99*, pages 175–181, New York, NY, USA, 1999.

[20] S. Lin and J. Ho. Discovering informative content blocks from web documents. In *Proceedings of SIGKDD '02*, pages 588–593, New York, NY, USA, 2002.

[21] L. Liu, C. Pu, and W. Han. Xwrap: An xml-enabled wrapper construction system for web information sources. In *Proceedings of ICDE '00*, pages 611–621, 2000.

[22] C. Mantratzis, M. Orgun, and S. Cassidy. Separating xhtml content from navigation clutter using dom-structure block analysis. In *Proceedings of HYPERTEXT '05*, pages 145–147, 2005.

[23] M. Marek, P. Pecina, and M. Spousta. Web page cleaning with conditional random fields. In *Proceedings of the Web as Corpus Workshop (WAC3),Cleaneval Session*, 2007.

[24] D. Pinto, M. Branstein, R. Coleman, W. B. Croft, M. King, W. Li, and X. Wei. Quasm: A system for question answering using semi-structured data. In *Proceedings of JCDL '02*, pages 46–55, 2002.

[25] A. F. R. Rahman, H. Alam, and R. Hartono. Content extraction from html documents. In *WDA2001*, pages 7–10, 2001.

[26] R. Song, H. Liu, J. Wen, and W. Ma. Learning block importance models for web pages. In *Proceedings of WWW '04*, pages 203–211, New York, NY, USA, 2004.

[27] T. Weninger, W. H. Hsu, and J. Han. Cetr - content extraction via tag ratios. In *Proceedings of WWW '10*, pages 971–980, New York, NY, USA, 2010.

[28] L. Yi, B. Liu, and X. Li. Eliminating noisy information in web pages for data mining. In *Proceedings of SIGKDD '03*, pages 296–305, New York, NY, USA, 2003.