TSINGHUA SCIENCE AND TECHNOLOGY
ISSN 1007-0214 05/18 pp256-264
Volume 17, Number 3, June 2012

Combining a Segmentation-Like Approach and a Density-Based Approach in Content Extraction*

Shuang Lin, Jie Chen, Zhendong Niu**

School of Computer Science, Beijing Institute of Technology, Beijing 100081, China

Abstract: Density-based approaches in content extraction, whose task is to extract contents from Web pages, are commonly used to obtain page contents that are critical to many Web mining applications. However, traditional density-based approaches cannot effectively manage pages that contain short contents and long noises. To overcome this problem, in this paper, we propose a content extraction approach for obtaining content from news pages that combines a segmentation-like approach and a density-based approach. A tool called BlockExtractor was developed based on this approach. BlockExtractor identifies contents in three steps. First, it looks for all Block-Level Elements (BLE) & Inline Elements (IE) blocks, which are designed to roughly segment pages into blocks. Second, it computes the densities of each BLE&IE block and its element to eliminate noises. Third, it removes all redundant BLE&IE blocks that have emerged in other pages from the same site. Compared with three other density-based approaches, our approach shows significant advantages in both precision and recall.

Key words: content extraction; segmentation; density-based approach

Introduction

The contents of Web pages are the primary focus of Web mining applications, such as the use of search engines and topic detection. Unfortunately, Web pages such as the ones shown in Fig. 1 are always surrounded by many noises. According to Gibson et al., about 40%-50% of the data on the Web are noises[1]. In addition to noises, the heterogeneity of pages and demands for automation and efficiency make it difficult to extract contents from pages. If Web pages were written according to a common template, we could easily extract content simply by writing a regular expression. However, this quickly becomes impractical when dealing with hundreds of Web pages that are generated from different templates.

One way to complete this task is to segment Web pages into blocks and then detect the contents from the block set. In this way, Web pages are represented in HTML Document Object Model (DOM). According to HTML specifications and programming practices, in a node tree, related data will always be gathered as brothers and unrelated data will be separated. In other words, contents are likely to be placed in the same branch and the same layer of the node tree, while noises are likely to be separated. This makes it possible to extract content by Web page segmentation[2-4]. Once pages have been segmented into blocks, the extraction can be done in each block, and the precision of extraction will be increased.

Another way to extract contents is based on the fact that contents always contain large numbers of characters and need relatively few characters to depict their

Received: 2012-04-26; revised: 2012-05-14
* Supported by the Program for Beijing Municipal Commission of Education (No.1320037010601), the 111 Project of Beijing Institute of Technology, and the National Key Basic Research and Development (973) Program of China (No. 2012CB7207002)
** To whom correspondence should be addressed.
E-mail: zniu@bit.edu.cn; Tel: 86-13693063092

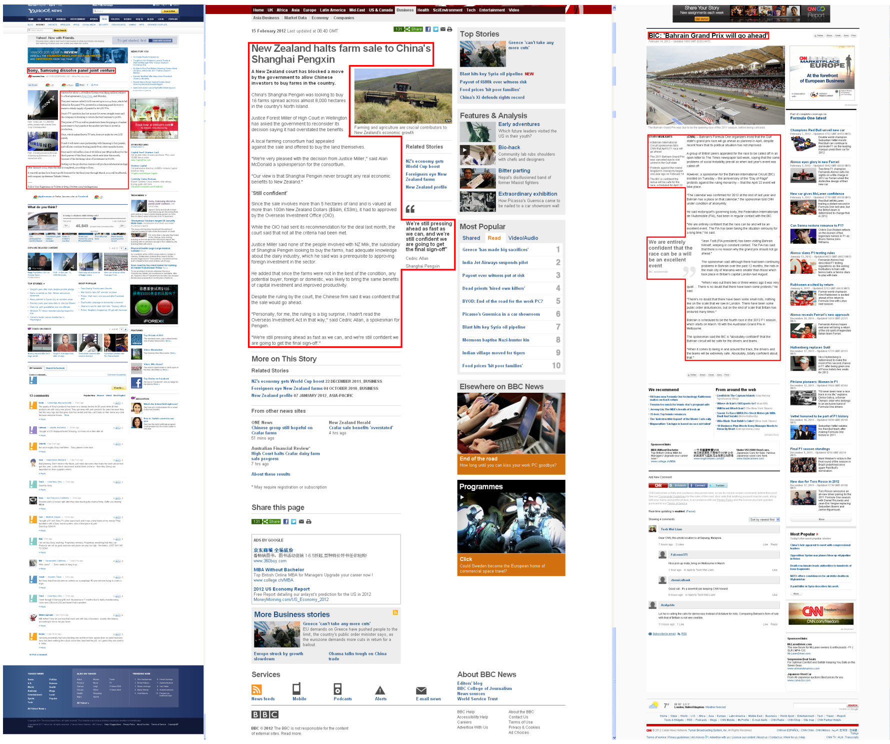

Fig. 1 Web pages from Yahoo!, BBC, and CNN with contents in red boxes

tags, while noises always contain only a few characters but need relatively large numbers of characters to depict their tags. The approaches based on this observation are known as density-based approaches[5-7]; they are always easy to implement, and they are quite efficient as well. However, they have trouble managing pages with short contents and long noises.

In this paper, we propose a content extraction approach that combines a segmentation-like approach and a density-based approach. Based on the heuristics of Block-Level Elements (denoted as BLE) and Inline Elements (denoted as IE), our approach looks for BLE&IE blocks and eliminates noises by calculating the density and detecting the redundancy of each BLE&IE block. Although the idea of obtaining BLE&IE blocks from node trees may sound like segmentation, in this approach, formal segmentation approaches are omitted to increase efficiency. Actually, only the subtrees in a node tree that contain text will be captured. In addition, BLE&IE blocks are designed to help identify content by discovering subtrees that are suitable for applying a density-based approach and removing redundant blocks. Since no machine learning technology is used, our approach does not require training or manual labeling of data. In addition, it is easy to implement and can be executed quickly. As far as we know, no one has ever tried pre-segmentation to improve the performance of density-based approaches. We conducted experiments with several naïve density-based approaches and our hybrid approach, and then compared the results.

1 Related Work

Content extraction from HTML documents was first introduced to meet the demand for obtaining information from the Web for Personal Digital Assistants (PDAs) and cellular phones[8]. Nowadays, as Web data mining applications have continued to develop, content extraction is often used to get data from pages while eliminating noises. Over the past several years, researchers have proposed many approaches for content extraction.

In the early years, people treated Web pages as plain text. Finn et al.[9] split pages into tokens of tags and texts before extraction. As far as we know, DOM was first used in the field of content extraction by Gupta et al.[10], who developed a system to convert Web pages into DOM and obtain contents using filters. Adelberg[11] developed a tool called NoDoSE, which requires users to decide what to extract. Gupta et al.[12] designed Crunch, a framework combining different content

extraction heuristics. Since different heuristics are suitable for different kinds of Web pages, Crunch classifies Web pages and associated pages into different "genres". It then employs previously determined settings that are appropriate for specific genres to extract contents. Gottron[13] extended this approach and designed the CombinE system. The most significant improvement was that the CombinE system made it possible to get optimized settings by training with data automatically. However, these approaches lack automation, which plays an important role in many Web applications that deal with massive numbers of pages.

Template detection was another technology used to obtain contents by detecting a common template among pages from the same site. Reis et al.[14] used tree edit distance to get the template. Yi et al.[15] proposed a structure called style tree to obtain the template of pages; by mapping pages to the Site Style Tree (SST), noises could be detected and eliminated. Gottron showed that any template detection algorithm could be used to extract content by building suitable training sets[16]. However, if a Web site changes its template, template detection-based approaches will fail because they are based on the assumption that pages from same site share the same template.

Segmentation-based approaches try to segment Web pages and identify contents by determining the informative block. Lin and Ho[17] utilized tag clues, especially table tag, to segment Web pages and then employed entropy to obtain the contents. Similarly, Debnath et al.[18] utilized more tag clues to split Web pages into blocks and then used the k-means algorithm to detect content. However, the clues of some specific tags cannot remain robust enough as time passes and as preferences of tag usage change. VIPS is a vision-based page segmentation algorithm that relies on page layout features such as width, height, font, color, etc. Lei et al.[2] employed VIPS to extract content, and Song et al.[19] proposed a method for calculating block importance based on VIPS. Nonetheless, VIPS uses style information to segment pages. In other words, this approach requires style sheets to be downloaded and parsed, which will significantly affect system efficiency.

Alexjc[7], Zhou et al.[5], and Sun et al.[6] all employed density-based approaches to extract content from Web pages. Alexjc[7] regarded density as the ratio of bytes of text to the bytes that were needed to encode the text line, including both text and tags. Zhou et al.[5] extended this approach by pre-processing the page as a paragraphed text string directly, extracting the content by analyzing the word count of each paragraph and comparing it to the maximum word count. Sun et al.[6] utilized DOM to calculate density; he regarded density as the ratio of bytes of text to tag count in the DOM subtree. The problem is that density-based approaches cannot effectively manage pages that contain short contents (e.g., subtitles) and long noises (e.g., copyrights and comments). Even though Zhou et al.[5] used smoothing to address this issue and Sun et al.[6] proposed DensitySum to solve this problem, neither one of these approaches was particularly efficient.

2 Our Approach

2.1 Related concepts

2.1.1 HTML DOM

According to the specifications developed by the World Wide Web Consortium (W3C), HTML DOM defines a standard way to access and manipulate HTML documents, and it views an HTML document as a tree structure called a node tree. Everything in an HTML document is a node. The whole document is regarded as a document node with HTML elements as element nodes, text as text nodes, comments as comment nodes, and HTML attributes as attribute nodes. Figure 2 shows an example of an HTML document and its node tree.

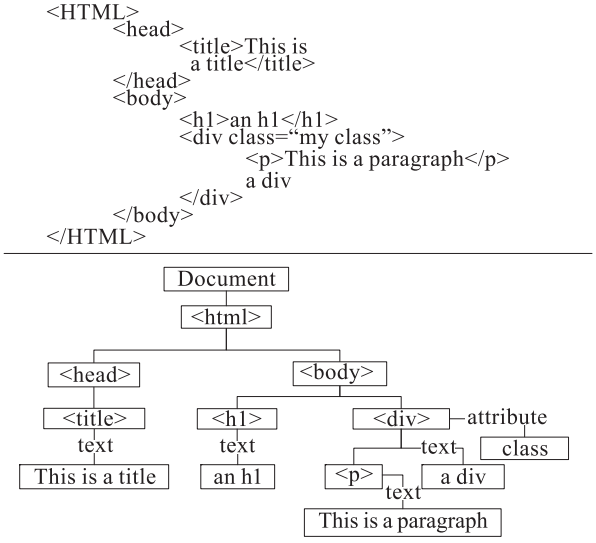

Fig. 2 An HTML document (above) and its node tree (below). The node tree presents nodes in the document and the connections between them.

2.1.2 Block-level elements and inline elements

An element is the basic component of an HTML document, and each one corresponds to an element node in the node tree. Elements that are displayable can be rendered in browsers as either BLE or IE.

• BLE are elements that are displayed as block margins in a new line with independent height and width. They can contain text, IE, or other BLE.

• IE are elements that are displayed inline with margins, width, and height inherited from BLE. They can only contain text or IE.

Element <p> and headings like <h1>, <h2>, etc. that create texts are called basic texts in BLE. And elements like <dl>, <ol>, and <li> that define a list are called lists. Elements like <blockquote>, , <address>, and <hr> are BLE, too. The most common IE in HTML documents are <a>, , , etc.

2.2 BLE&IE blocks

It is feasible to segment Web pages and determine contents by determining the informative block. One way to segment Web pages is to use heuristic rules of HTML tags. However, the heuristic rules used in many studies are based on specific tags, and this will decrease the universal applicability of these approaches. Another way is to utilize visual heuristics. However, visual-based segmentation also requires style sheets to be downloaded and parsed, which will significantly affect efficiency. Although there are many disadvantages in segmentation approaches, we find that after segmentation, contents or noises will be gathered, from which density-based approaches can benefit.

We designed BLE&IE blocks to work in a similar way as segmentation. They are designed to discover subtrees and apply a density-based approach and remove redundancy. To overcome the previously mentioned shortcomings of other approaches, we increased the heuristics granularity from specific tags to BLE and IE, ignoring visual heuristics. In an HTML document, BLE are always displayed as block margins in a new line, while IE are displayed inline. Moreover, the properties of BLE and IE will not be changed by Cascading Style Sheets (CSS), nor easily modified by W3C as time passes. We find that these clues can help to "segment" pages roughly, and even though they are not tremendously effective for the task of traditional segmentation, they are sufficient for finding blocks used for extraction. As a result, we use the term "segmentation-like approach". By observing large numbers of news Web pages, we find that: (1) both IE and text correspond to contents and noises; (2) the composition of consecutive IE and text in the same layer always refers to a whole; (3) since BLE are displayed as block margins in a new line, they always have a weak relationship with IE in the same layer, except for certain special BLE; (4) BLE with text only are special BLE (denoted as q); (5) BLE with IE and text directly as their children are special BLE (denoted as p); (6) list elements (denoted as l) in BLE always stay together, as do basic text elements (denoted as t). According to these observations, the definition of a BLE&IE block is as follows:

Definition 1 A BLE&IE block is a labeled ordered tree that is converted from a node tree by some specific operations (explained below); the root of a BLE&IE block is a BLE; the arrangement of the direct children of the tree root matches the regular expression (IE|text)*p?q*|l*|t*.

According to our definition, it is clear that the root of the BLE&IE block will be located near the leaves of the DOM tree since our definition focuses more on IE and text. This is because BLE&IE blocks are designed for density-based extraction.

A BLE&IE block can be a subtree from a node tree, but not all BLE&IE blocks are subtrees from a node tree. They are converted from a node tree, whose subtree may correspond to many BLE&IE blocks. In general, we can convert a node tree to BLE&IE blocks as follows:

• **Remove**. Nodes like script and comments should be removed straight away.

• **Indent**. Indenting all attribute nodes to their element node as a whole is done before other operations.

• **Cut**. Subtrees from a node tree do not always meet the definition, but they are still able to get BLE&IE blocks by cutting the right-most subtrees to meet the definition.

• **Reconstruct**. The remaining subtrees should be reconstructed into a new tree with a copy of their parents as roots.

• **Recursive**. When a subtree or a newly reconstructed tree cannot meet any definition after cutting, the algorithm must be recursive.

Figure 3 shows a subtree from the node tree and its

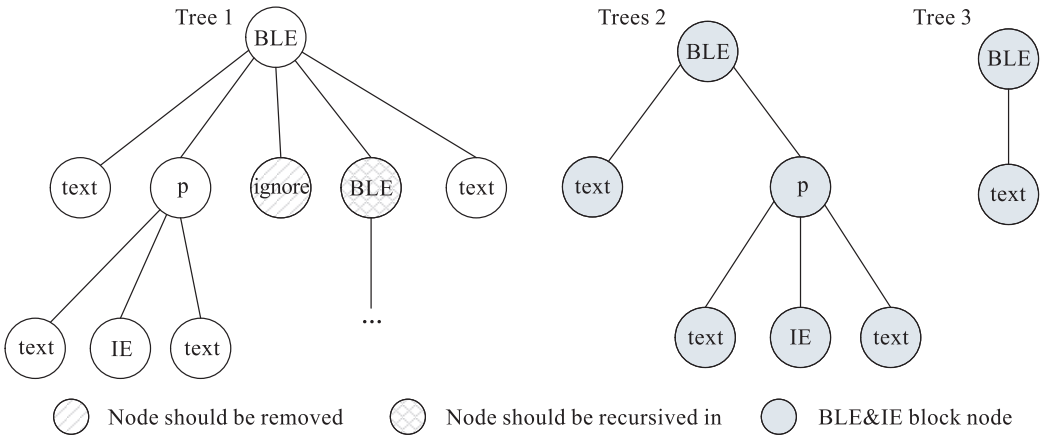

Fig. 3 An example of converting a subtree into BLE&IE blocks. Tree 1 is the original subtree, while Tree 2 and Tree 3 are BLE&IE blocks.

derived BLE&IE blocks. Algorithm 1 shows the process in detail.

Algorithm 1: Pseudo code of DOM2Blocks(*N*)

```
1.  Input: node N in node tree
2.  Output: blocks in node N
3.  Array S
4.  for child n in N:
5.      if n is script or comment:
6.          remove n from N and continue;
7.      elif n is attribute node:
8.          indent n to N as a property;
9.      else:
10.         identify the type of n and push it into S;
11.         if S don't meet (IE|text)*p?q*|l*|t*:
12.             pop n from S;
13.         cut the rest from N and get a BLE&IE
                block according to S;
14.             reconstruct the rest with a copy of
                    N as root;
15.             if n can't meet rules singly:
16.                 DOM2Blocks(n);
17.             else:
18.                 push n into S;
```

2.3 Using density to extract contents

In HTML documents, contents always contain large numbers of characters and need comparatively fewer characters to describe their tags, while texts in noises always contain small numbers of characters and need comparatively more characters to describe their tags. Based on this fact, many density-based approaches for content extraction have been developed. In this paper, we rely on the following definitions:

• **TagLength** (denoted as TgL): the length of a tag, including tag name, attributes, and the sum of these in all its child nodes. It can be calculated by:

$$\mathrm{TgL} = \mathrm{LofTag} + \mathrm{LofAttr} + \sum_{i\in\Omega}(\mathrm{LofTag}_i + \mathrm{LofAttr}_i) \quad (1)$$

where LofTag is the length of the tag name, LofAttr is the length of attributes, and Ω refers to the child node set.

• **TextLength** (denoted by TtL): the length of text in a tag (LofText) and the sum of it in all its child nodes ($\mathrm{LofTest}_i,\ i \in \Omega$). It can be calculated by:

$$\mathrm{TtL} = \mathrm{LofText} + \sum_{i\in\Omega}\mathrm{LofText}_i \quad (2)$$

With the definition of TgL and TtL, we can define the density of a node or BLE&IE block as follows:

Definition 2 The density (denoted as ρ) of a node or a BLE&IE block is the ratio of TtL to TgL:

$$\rho = \frac{\mathrm{TtL}}{\mathrm{TgL}} \quad (3)$$

Note that we are only concerned about the densities of the BLE&IE blocks and the BLE in them. Since we treat text nodes as nodes without densities and they always appear with IE among them, in consideration of text coherence, it is better to treat IE as nodes without densities, too. After obtaining the density of the BLE node in each BLE&IE block, we use a fixed threshold to determine content candidates, and only those nodes with densities and their ancestors' densities (if they exist) that are greater than the threshold will be regarded as content candidates. In other words, if the density of the root of a BLE&IE block is below the threshold, the whole BLE&IE block will be regarded

as a noise. The simple process of obtaining the density of each node in a BLE&IE block and determining content candidates can be seen in Algorithm 2.

Algorithm 2: Pseudo code of get_content_candidates (*N*)

```
1.  Input: node N in BLE&IE Block
2.  Output: {TtL, TgL}
3.  init {TtL, TgL} with data from N;
4.  if N is a BLE:
5.    for each child n in N:
6.      {TtL', TgL'}= get_ content_candidates (n);
7.      increase {TtL, TgL} by {TtL', TgL'};
8.      get the density of n according to {TtL', TgL'};
9.        if density of n is less than Threshold:
10.           remark n as a noise;
11.   return {TtL, TgL};
```

2.4 Redundancy removal

When referring to pages from the same site, we may find that some noises that are regarded as redundant are shared by other pages. Although it is possible to eliminate noises by detecting redundancy, it is difficult to correctly detect redundancy over an entire Web site. For example, we may find that a title that is considered to be a piece of content may repeatedly exist in another page from the same site as related news, but should actually be regarded as noise. However, things will get better if the page is segmented, in which we can detect redundancy in terms of blocks by evaluating the tree edit distance[20] of the blocks. As a result, the precision of redundancy detection can be increased dramatically.

3 Experiments

We developed a content extraction tool for news Web pages called BlockExtractor (BE) based on our approach. It identifies contents in three steps. First, BlockExtractor looks for all BLE&IE blocks that are designed for density-based extraction in Web pages. Then, BE computes densities of each BLE&IE block to eliminate noises. Third, BE removes all redundant BLE&IE blocks that have emerged in other pages from the same site.

3.1 Data set

The data set used in our experiment consists of news pages from Yahoo!, CNN, BBC, and the New York Times (NY Times). We randomly downloaded 100 pages from each site, and the contents of these pages were extracted manually and saved as standard. In order to make the data set more varied, we tried our best to get different kinds of news from these sites (e.g., sports, current affairs, technology, education, etc.).

3.2 Performance metrics

Precision (p), recall (r), and F_1-measure (F_1) were used to evaluate the experimental results and compare the performance of each approach. Precision is the fraction of extracted texts that are contents, and recall is the fraction of the contents that is extracted. The F_1-measure, which is the weighted harmonic mean of precision and recall, gives each of them equal weight. These metrics can be calculated by the following equations:

$$p = \frac{f(\text{Rslt},\text{Strd})}{\text{length of Rslt}} \tag{4}$$

$$r = \frac{f(\text{Rslt},\text{Strd})}{\text{length of Strd}} \tag{5}$$

$$F_1(p,r) = \frac{2pr}{p+r} \tag{6}$$

where Rslt represents the result of extraction, Strd represents the standard, and the f function is used to get the longest common subsequence[21] of Rslt and Strd.

3.3 Results

Since the threshold in our approach is a previously determined fixed value, there is a tradeoff between how high the threshold should be in order to balance precision, recall, and the F_1-measure. Figures 4 and 5 show the precision, recall, and the F_1-measure as the threshold of BlockExtractor increases. Note that when the threshold is 0, BlockExtractor will achieve performance with maximum recall and minimal precision. When the threshold increases, recall decreases, precision increases, and F_1 has a maximum value of about 1.6. According to Figs. 4 and 5, it appears that when the threshold is between 1.3 and 1.7, BlockExtractor has the most balanced performance. As a result, all experiments for BlockExtractor utilize a threshold of 1.5. Table 1 presents the detailed performance of BlockExtractor for each Web site when the threshold is 1.5.

For the purpose of comparison, we implemented the approach proposed by Alexjc[7], the main text extraction for single topic Web pages proposed by Zhou et

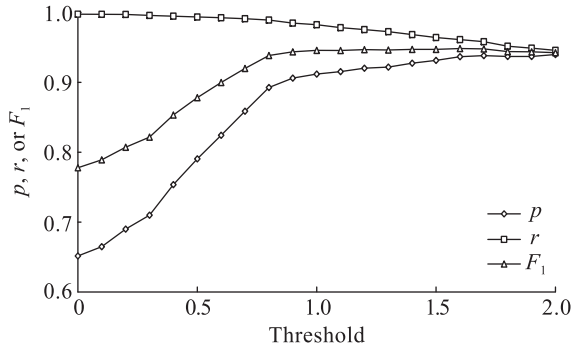

Fig. 4 Precision, recall, and the F_1-measure as the threshold of BlockExtractor increases from 0 to 2 (in increments of 0.1)

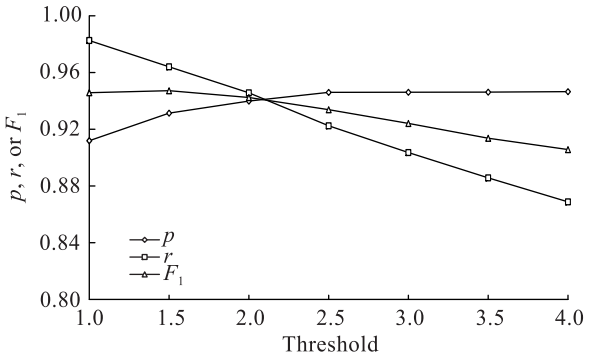

Fig. 5 Precision, recall, and the F_1-measure as the threshold of BlockExtractor increases from 1 to 4 (in increments of 0.5)

Table 1 Results of BlockExtractor when its threshold is 1.5

	Precision	Recall	F_1-measure
Yahoo!	0.9587	0.9566	0.9576
CNN	0.8915	0.9703	0.9293
BBC	0.9656	0.9812	0.9734
NY Times	0.9097	0.9482	0.9286
Average	0.9314	0.9640	0.9472

al.[5], and CETD-DS proposed by Sun et al.[6] The results of these alternative approaches, as well as those of BlockExtractor, are presented in Table 2.

3.4 Discussion

Since BlockExtractor is an extractor with a pre-determined threshold, it is important to identify the range within which performance is most balanced. Even though it is inevitable that precision increases and recall decreases when the threshold increases (as shown in Figs. 4 and 5), when the threshold is between 1.3 and 1.7, BlockExtractor achieves the most balanced performance, so the threshold was set at 1.5. This conclusion is based on the following assumptions: that

Table 2 Comparison between the three alternative approaches and BlockExtractor

	Approach	Precision	Recall	F_1-measure
Yahoo!	Zhou et al.[5]	0.1994	0.2244	0.2111
	Sun et al.[6]	0.3895	0.9357	0.5501
	Alexjc[7]	0.4820	0.8900	0.6254
	BE	**0.9587**	**0.9566**	**0.9576**
CNN	Zhou et al.[5]	**0.9201**	0.9731	**0.9458**
	Sun et al.[6]	0.7572	**0.9908**	0.8584
	Alexjc[7]	0.8284	0.9771	0.8967
	BE	0.8915	0.9703	0.9293
BBC	Zhou et al.[5]	0.4690	0.4107	0.4380
	Sun et al.[6]	0.5789	**0.9926**	0.7313
	Alexjc[7]	0.6122	0.9630	0.7486
	BE	**0.9656**	0.9812	**0.9734**
NY Times	Zhou et al.[5]	0.9184	0.9364	0.9273
	Sun et al.[6]	0.8358	**0.9682**	0.897
	Alexjc[5]	0.8373	0.9516	0.8908
	BE	**0.9097**	0.9482	**0.9286**
Average	Zhou et al.[5]	0.6267	0.6362	0.6301
	Sun et al.[6]	0.6404	**0.9718**	0.7592
	Alexjc[7]	0.6900	0.9454	0.7904
	BE	**0.9314**	0.9640	**0.9472**

Figs. 4 and 5 can correctly depict the curves of precision, recall, and F_1 based on varying thresholds in general; and that the most balanced performance will have a maximized F_1. To prove the first point, we experimented with as many thresholds as possible. Certain empirical clues tell us that as threshold increases, precision must go up, recall must decrease, and F_1 increases to its maximum value and then decreases. All of these ensure that Figs. 4 and 5 can correctly depict the curves of precision, recall, and F_1 based on varying thresholds in general. As for evaluating the most balanced performance, we utilized the F_1-measure because it is the weighted harmonic mean of precision and recall, and it considers both precision and recall in equal measure.

Table 2 shows the performance of the other density-based approaches and BlockExtractor. First, the results show that our approach outperforms the other three approaches on average, achieving the best precision, the best F_1, and the second-best recall. This is because the other three density-based approaches cannot effectively manage pages with long noises and short contents, but BlockExtractor can since BLE&IE Blocks hold related noises or contents together and

redundancy can be detected at the block unit. Second, the density-based approaches all achieved satisfying recall except for Zhou et al.'s approach[5], which performed terribly on Yahoo! and BBC. Once density is used to extract contents, those who want to extract the maximum amount of content may adjust either the threshold or the algorithm that is used to determine the threshold. In this way, density-based approaches are likely to achieve satisfying recall. Third, the other naïve density-based approaches showed similar performance overall, highlighting the limitations of such naïve density-based approaches, as well as the need to utilize other methods.

4 Conclusions

In this paper, we proposed a content extraction approach that combines a segmentation-like approach and a density-based approach. In our approach, we designed structures called BLE&IE blocks to gather related noises or contents. Next, we used this density-based approach and redundancy removal to obtain the final content. Based on our approach, a tool called BlockExtractor was developed. Experimental results on our data set showed that this novel approach was effective and robust compared to three other density-based approaches.

In our future work, we will redesign our approach to make it applicable to other kinds of Web pages such as blogs and forums. We also plan to adopt our approach for Web data mining applications.

Acknowledgments

The authors would like to thank Kunshan Wang for proposing to us the idea of designing a fully automatic content extractor. We also want to thank Yulong Shi for helping us gather data. Last, we are grateful to Yao Yao for his revision suggestions.

References

[1] Gibson D, Punera K, Tomkins A. The volume and evolution of web page templates. In: Proceedings of WWW'05. New York, NY, USA, 2005: 830-839.

[2] Lei F, Yao M, Hao Y. Improve the performance of the webpage content extraction using webpage segmentation algorithm. In: Proceedings of International Forum on Computer Science-Technology and Applications. Chongqing, China, 2009: 323-325.

[3] Debnath S, Mitra P, Giles C L. Automatic extraction of informative blocks from webpages. In: Proceedings of SAC'05. New York, NY, USA, 2005: 1722-1726.

[4] Cai D, Yu S, Wen J R, et al. VIPS: A vision based page segmentation algorithm. Technical Report MSR-TR-2003-79, Microsoft, 2004.

[5] Zhou B, Xiong Y, Liu W. Efficient web page main text extraction towards online news analysis. In: Proceedings of 2009 IEEE International Conference on e-Business Engineering. Piscataway, NJ, USA, 2009: 37-41.

[6] Sun F, Song D, Liao L. DOM based content extraction via text density. In: Proceedings of the 34th International ACM SIGIR Conference. Beijing, China, 2011: 245-254.

[7] Alexjc. The easy way to extract useful text from arbitrary HTML. http://ai-depot.com/articles/the-easy-way-to-extract-useful-text-from-arbitrary-HTML/. 2007.

[8] Rahman A F R, Alam H, Hartono R. Content extraction from HTML documents. In: Proceedings of WDA2001. Seattle, WA, USA, 2001: 7-10.

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

[10] Gupta S, Kaiser G, Neistadt D, et al. DOM-based content extraction of HTML documents. In: Proceedings of the 12th International Conference on World Wide Web. Budapest, Hungary, 2003.

[11] Adelberg B. Nodose—A tool for semi-automatically extracting semi-structured data from text documents. In: Proceedings of SIGMOD'98. New York, NY, USA, 1998: 283-294.

[12] Gupta S, Kaiser G, Stolfo S. Extracting context to improve accuracy for HTML content extraction. In: Proceedings of WWW'05. New York, NY, USA, 2005: 1114-1115.

[13] Gottron T. Combining content extraction heuristics: The CombinE system. In: Proceedings of iiWAS'08. New York, NY, USA, 2008: 591-595.

[14] Reis D C, Golgher P B, Silva A S, et al. Automatic web news extraction using tree edit distance. In: Proceedings of the 13th International Conference on World Wide Web. New York, NY, USA, 2004: 502-511.

[15] Yi L, Liu B, Li X. Eliminating noisy information in web pages for data mining. In: Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. New York, NY, USA, 2003: 296-305.

[16] Gottron T. Bridging the gap: From multi document template detection to single document content extraction. In:

Proceedings of the IASTED Conference on Internet and Multimedia Systems and Applications 2008. Calgary, Canada: ACTA Press, 2008: 66-71.

[17] Lin S, Ho J. Discovering informative content blocks from web documents. In: Proceedings of SIGKDD'02. New York, NY, USA, 2002: 588-593.

[18] Debnath S, Mitra P, Giles C L. Identifying content blocks from web documents. In: Proceedings of International Syposium on Methodologies for Intelligent Systems. New York, NY, USA, 2005: 285-293.

[19] Song R, Liu H, Wen J, et al. Learning block importance models for web pages. In: Proceedings of the 13th International Conference on World Wide Web. New York, NY, USA, 2004: 203-211.

[20] Tai K C. The tree-to-tree correction problem. *Journal of the ACM* (*JACM*), 1979, **26**(3): 422-433.

[21] Hirschberg D S. Algorithms for the longest common subsequence problem. *Journal of the ACM* (*JACM*), 1977, **24**(4): 664-675.