Primary Content Extraction with Mountain Model*

Lidong Bing, Yexin Wang, Yan Zhang†
Key Laboratory of Machine Perception (Ministry of Education)
Peking University, 100871, Beijing, China
{bingld, wangyx, zhy}@cis.pku.edu.cn

Hui Wang
Songchang Technology Inc.
Beijing, China
hui.wang.cn@gmail.com

Abstract

It is necessary to eliminate cluttered information in Web pages, such as navigation bars, related readings, copyright notices, since they can cause additional burden to search engines. In this paper, a Web page is treated as a sequence of content cells, where each cell owns its score according to our Mountain Model. Primary content cells are distinguished from those cluttered content cells by the features processed only by primary cells. A universal classifier is trained based on these features for a global utility. To make it more precise, we also provide a site-oriented classifier. An algorithm is thus schemed out for primary content extraction based on Mountain Model. Experimental results show that our model works with both accuracy and time efficiency compared with the existing models.

1. Introduction

In Web pages, cluttered information includes navigation bars, related readings, advertisement links, copyright notices, responsibility statement and time-stamps. Such information items are functionally useful for human viewers and necessary for the Web site owners, however, cluttered message can cripple the performance of many modules of search engines, including the index, ranking function, summarization, duplicate detection, etc. For example, in one data set of our experiment, 46.6% of the 821 pages' visible information is cluttered information. What burden will search engines suffer from these clutters? No doubt that it will bring about a lot of difficulties. So for the sake of improving the performance of search engines, it is highly necessary to filter out the cluttered content from the primary content.

However, as far as we have known, there is no quite potent solution to handle this problem with both high accuracy and efficiency. There are mainly two obstacles in the way. Firstly, to be up against so heterogeneous a Web, any attempt to seek a method which works both accurately and universally is not an easy task. Secondly, the method's time complexity must be acceptable when applied to process a large amount of pages. Page structure based methods [17][14][8] are required to adapt themselves to different Web sites' template structures since different sites have their own templates in publishing pages, which is very difficult for machines, as we know. Another disadvantage of the page structure based methods is that they must consider the site as a whole while processing, because only several pages cannot reveal the structure's law in the site, it must bring in a waste of both time and memory space. A case in point is that, DOM tree [1] structure are widely used to analyze Web pages and extract primary contents[5][15][3]. Confessedly it spends too much time on both building the DOM tree and navigating in it, so time cost is intolerable when these methods are employed to deal with large amounts of pages.

In order to overcome the above obstacles and eliminate the limitations, we propose a novel model to present a Web page. Visible information of a page is scattered in different bars according to its position in the HTML code when we use a standing bar to figure its existence, and the bar's height to denote its score. Since the list of these bars composes a shape like an undulating mountain, we call it a Mountain Model (MM). When we establish an MM to present a page, it is necessary to draw a clear picture of the differences between informative fragments and uninformative ones based on both structure-independent and site-independent features of informative fragments. To distinguish the topic related informative fragments from the cluttered information, a learning-based SVM classifier is used.

The rest of the paper is organized as follows. In Section 2, we review related work. The Mountain Model is described in Section 3. Section 4 presents implementation for identifying primary content of a Web page based on the Mountain Model, and Section 5 shows the performance of this model. Finally we draw the conclusion in Section 6.

*Supported by NSFC under Grant No.60673129, 60773162 and 60573166, Beijing Natural Science Foundation under Grant No.4073034
†Corresponding author

2. Related Work

According to the work of Yi *et al.*[17], in most studies of Information Retrieval, Web page data cleansing can be grouped into two categories: global-scale data cleansing and local-scale data cleansing. The work of Global-scale data cleansing includes PageRank[2], HITS[13], TrustRank[12], and they are based on the hyperlink structure of Web pages, to upgrade the high-quality pages and degrade the low-quality ones.

In local-scale Web data cleansing, many researchers have considered using the tag information and dividing a page on the basis of the type of tags[16]. Cai *et al.*[5, 4] use the layout structures to build the visual structure of a Web page and fulfill the partitioning task in terms of the visual structure. After a Web page is partitioned into several blocks, algorithms based on learning mechanisms[15] or based on hyperlink structure [3] can be performed to locate the important blocks or cleanse the unimportant ones. I. Chibane et al. also present their work based on visual structure [7], and they assume that every visual block denotes a topic.

Besides using information inside a Web page, researchers have also tried to find the common style of noisy data inside a Web site, which is called Site Style Tree by Yi *et al.*[17]. With this method they denote Web pages in a site into main content blocks and noisy blocks. Other site-oriented methods include [11, 10, 14, 8].In [11], an advertisement server blacklist is used to eliminate the advertisement. A genre-based clustering approach is employed in [10], where different genres have their own filter configurations. In [14], entropy of a page block is computed based on the key words, then people dynamically select the threshold of entropy that partitions blocks into either informative or redundant. A tag-set is used in [8] to divide a page into blocks, and then IBDF is employed to denote the blocks' importance, similar to IDF for words. Page level template detection is studied in [6], where the authors investigate the pages' features, and use these features to score the DOM tree nodes. After isotonic smoothing is done on classifier scores, page level templates are generated accordingly.

3. Mountain Model

Definition 1 (Primary Content): In a Web page, the primary content is the most important and useful part in terms of semantic meaning.

Web page document is a semi-structured data presentation form, where visible information is what end users can see, including primary content and cluttered information. Visible information is usually surrounded by invisible part, which includes HTML code and other scripts.

3.1. Information Cell

Definition 2 (Information Cell): In Web page source codes, a visible information segment that inclosed in '$>$' and '$<$' tightly is called an Information Cell (IC).

All visible information of a Web page belongs to a certain IC, no matter it is primary content or cluttered. In our Mountain Model (MM), a Web page is treated as an IC sequence after eliminating all invisible codes. Thus primary content extraction from a page is transformed into primary ICs identifying in an IC sequence. Figure 1 presents a Web page's IC sequence, where x-coordinate is the serial number of Cells, and y-coordinate denotes Cells' score. In this example page, the primary content scatters from 48 to 112 approximately, and at the end of the sequence there are several high score Cells, which contains copyright declaration.

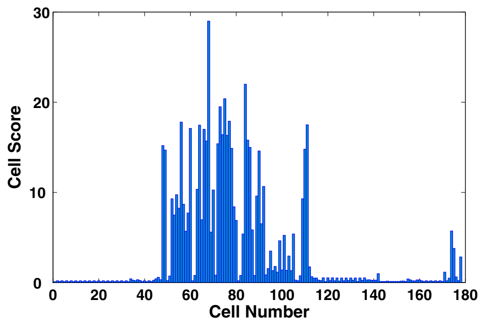

Figure 1. A Web page's mountain model presentation.

The Cell feature selection should play a part in distinguishing the primary Cells and cluttered Cells. In most cases, primary content has higher cohesion and its Cells are content rich ones, while anchor text seldom appears in it. Some of cluttered information are hyperlink styled, such as navigation bars, related readings and etc, so the visible information of them is anchor text, which seldom contains punctuation, especially full stop. Other cluttered information are plain texts, such as copyright and time-stamps. This kind of cluttered information is often relatively short and surrounded by a large amount of source codes. The features listed in Table 1 are used to describe a single Cell.

In Table 1, text length denotes a Cell's visible character number, and punctuation number represents how many sentence segmentation punctuations the Cell contains. Whether the Cell's information is full stopped is indicated with a Boolean symbol S. Symbol A indicates whether it is anchor text. Visible Ratio is the proportion occupied by the visible content in a Cell's source code span. A Cell's source code span is the code length from the center of its left neighbor's right boundary '$<$' and it owns left boundary '$>$' to

Table 1. Cell Features

Name	Symbol	Function
Text Length	*L*	Both
Punctuation Number	*P*	Link Style
Whether Full Stopped	*S*	Link Style
Visible Ratio	*V*	Both
Whether Anchor text	*A*	Link Style

the center of itself and its right neighbor. The function column in Table 1 denotes the feature can be applied to both of two clutter styles or only one. The Cell scoring function is:

$$CS = (L/\gamma + P)(1 + \alpha S)(1 + \beta A)V \tag{1}$$

where α , β, and γ are predefined constants. Normally, we set α = 0.5, β = -0.5, by which we mean that a full stopped Cell's score is upgraded 50%, while an anchor text should be degraded 50% respectively. Constant γ depends on the language of the Web page. For example, a sentence in Chinese contains 12 Chinese characters on average, γ is 12 in Chinese language environments; while in English, about 15 words in a sentence on average, and 5 letters in a word on average, so γ is set to 75 in English. If the language's statistics information is unkown, γ can be assigned 1.

3.2. Smoothing Function

According to the probability distribution in Figure 1, the primary Cells' neighbors are more likely to be primary, and cluttered Cells' are more likely to be cluttered, so the effects coming from a Cell's neighbors should be considered in identifying the Cell primary or cluttered. For example, the score of primary Cell 81 is low, however, if its surroundings are considered together, it can also be judged as primary. For the same reason, scores of 174 and 175 should be dragged down by their surroundings. Therefor, we may safely draw the conclusion that high score Cells upgrade their neighbors, whereas the low ones degrade their neighbors. This process is called *smoothing*:

$$CSS_n = CS_n + \frac{\sum_{i=n-\lfloor sw/2 \rfloor, i \neq n}^{n+\lfloor sw/2 \rfloor}(CS_i - AVG_n) \times w_i}{sw - 1} \tag{2}$$

where CSS_n is the score of Cell *n* after smoothing, CS_n is the original score, sw is the smoothing window size, AVG_n is the average original score of the Cells in the window, w_i is a weight factor assigned according to Equation 3.

$$w_i = \begin{cases} 1 - |n-i| \times 0.2 & \text{if |n-i|} \leq 4, \\ 0.1 & \text{if |n-i|} > 4. \end{cases} \tag{3}$$

When we only consider the neighbor Cells' effect, meanwhile neglecting the source code distance (SCD), side-effect appears. Two neighboring Cells' SCD is the code's length between the left one's right boundary '<' and the right one's left boundary '>'. The further two Cells' SCD is, the weaker effect they have on each other. To manifest two neighboring Cells SCD and separate them, some blank Cells are inserted. Since primary content often has high cohesion, very few blank Cells will be inserted. On the boundary of primary content and cluttered information, there are much source code and script normally, inserted blank Cells can prevent the primary Cells upgrading the cluttered ones to a certain extent. Obviously, we must implement blank Cell insertion before applying smoothing function.

3.3. Information Ridge

Primary information Cells have high scores and mass distribute in one or several segments in the MM, or Ridges.

Definition 3 (Information Ridge): Information Ridge is a sequence of Cells which satisfies the following three conditions: Firstly, at least one Cell's score is higher than *peak threshold*; Secondly, there is no such continuous subsequence exists, whose length is longer than a span window size and the containing Cells' scores are all smaller than *primary threshold*; Thirdly, the first and last Cells' scores are larger than *primary threshold*.

Primary threshold is the average score of all the Cells contained in a page. By expanding *primary threshold* for certain times, we get *peak threshold*. The multiplier is *peak factor*.

Some Web pages contain long cluttered information, such as responsibility statement, which is sometimes even longer than some short primary. So considering all the Ridges to be primary is too coarse. To distinguish primary Ridges and cluttered ones, we abstract four Ridge features.

Definition 4 (Ridge Features): A Ridge has four aspects of features: height, width, Ridge position (RP) and Cell position (CP), computed respectively by formula 4, 5, 6 and 7, where AVG_n is the average of Cells' scores in the nth Ridge, and AVG_{max} is the maximum one in all the Ridges. NoC_n is the number of Cells in the nth Ridge, and NoC_{max} is the maximum one in all the Ridges. NoR is the number of all the Ridges, FCN_n is the first Cell number of the nth Ridge, TC is the total number of Cells in the page, respectively. Observation shows that almost all primary Ridges have relatively larger height and width, and smaller RP and CP. The four features' values compose a vector, we call it feature vector(***FV***).

$$height = AVG_n/AVG_{max} \tag{4}$$

$$width = NoC_n/NoC_{max} \tag{5}$$

$$RP = n/NoR \tag{6}$$

$$CP = FCN_n/TC \tag{7}$$

```
Ridge Algorithm
Input: A Cells sequence C of a page
Output: Primary Content (PC)
begin
    RSet ← GetRidgeSet(C)
    for each r ∈ RSet do
        r.fv = GetFeatureVector(RSet,r)
        isPrimary ← Classify(r.fv)
        if isPrimary then
            PC = PC ∪ r.content
end
```

Figure 2. Ridge algorithm.

4. The Ridge Algorithm

There are three steps to extract primary content: obtain all Ridges based on MM, then compute $\boldsymbol{FV}(h, w, RP, CP)$ for each Ridge, finally distinguish primary Ridges from cluttered ones with a Support Vector Machine classifier. The pseudocode of Ridge Algorithm is shown in Figure 2.

In the $GetRidgeSet$ function, a Cell whose score is higher than *peak threshold* is detected at the beginning, and then from the backward and forward of the Cell, more Cells are added to the Ridge if they satisfy the Ridge conditions.

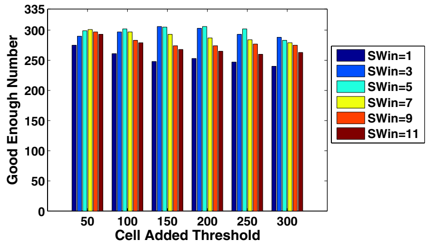

Figure 3. Effects of smoothing window size and Cell added threshold.

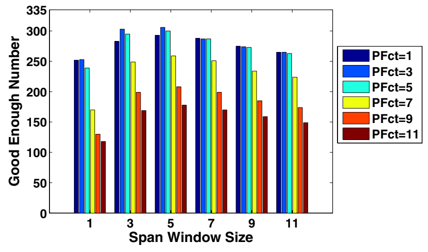

Figure 4. Effects of span window size and peak factor.

Table 2. Training Data Set

Domain	Sites
com	sina, 163, qq, msn, xinhuanet, sohu, tom, pcgames
edu	PKU, tsinghua, moe, neea
gov	pbc, gwytb, gov-money, cei
net	dinosaur, netmil, superarmy, 1n0, cyol
org	newssc, chinaschool, chinacourt
blog	blog.zol.com.cn, blog.qianlong.com, blog.edu.cn, 52blog.net, blog.qq.com, blog.163.com, blog.com.cn, blog.sohu.com, www.bokerb.com, blog.hexun.com

Continuously, all Ridges can be detected.

The training data set is listed in Table 2. To get a universal classifier (UC), we build a data set with a great diversity, which contains 1340 pages from 34 sites' 72 categories with 20 pages per category.

Four-fold cross validation is employed for training. We divide the training set into 4 equal parts. In each process, one part is used as the test set, and others as the training set. In the training set each page's Ridges are manually labeled as primary or cluttered, then we get a group of ***FVs***. Feeding this group features to an SVM training algorithm with RBF kernel to get a classifier, and then the classifier is used to judge the Ridges in the test set. Finally, four rounds' average performance under each parameter combination is computed for study of the effects of model parameters.

Figure 3 shows the number of pages with good enough F-Measure(given in next section) as a function of the Cell threshold and smoothing window size. In Figure 3, if the threshold is 50 and the SCD is 500, 10 blank Cells can be inserted. When the threshold is smaller, say 50, the larger smoothing windows have similar performance because too many blank Cells added. As the threshold grows, the performance of windows 9 and 11 becomes poorer. Smoothing window size 5 in our experiment outputs the best performance in each threshold. The threshold value is set to 200 when time efficiency is considered. The influence of span window size and peak factor on the performance of our method is shown in Figure 4. It is acceptable with span window size 5, and peak factor 3. We acquire all 1340 pages' ***FVs*** under the optimal parameters and label them manually, then feed them to SVM algorithm to train a UC.

5. Performance Evaluation

Precision is defined as the ratio of the number of correct sentences (primary content sentences) c extracted and the total number of sentences (all sentences suggested by an algorithm) t extracted ($p = c/t$). Recall is defined as

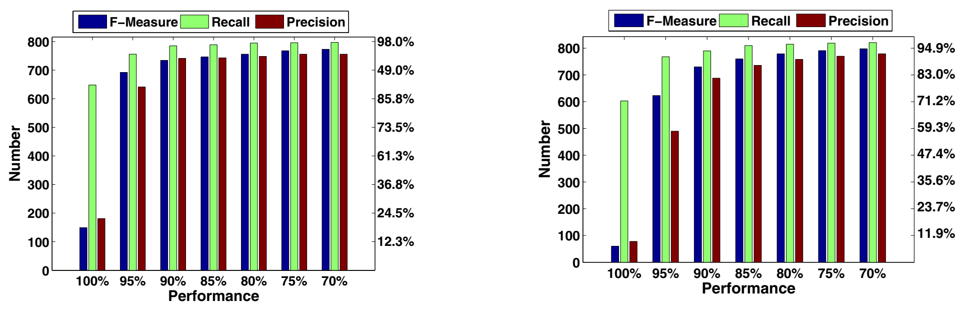

Figure 5. Performances on data sets U_1 and U_2.

the ratio of the number of correct sentences extracted and the desired number of correct sentences which includes the number of correct extracted sentences and the missed sentences m. That is $r = c/(c+m)$. F-Measure(FM) is defined as $FM = 2 \times pr/(p + r)$. If the FM value is larger than 0.9, the result is good enough. We use the Levenshtein Distance to instead the precision, recall and FM if the primary content cannot be segmented into sentences, such as name list pages. If a suggested primary Cell's content is anchor text and contains no punctuation, it is treated as a sentence.

Four data sets listed in Table 3 are employed to evaluate universal performance with UC. Diverse data sets U_1 and U_2 are results returned by *Google* with queries *Wanggang* and petroleum price. Web site data sets U_3 and U_4 are gained from *NDRC(www.ndrc.gov.cn)* and *Yahoo(www.yahoo.cn)*.

The performance of our algorithm on data set U_1 and U_2 is shown in Figure 5. On data set U_1, the number of pages with FM larger than 0.9 is 734, about 90% of the total. On data set U_2 this number is 730, 86% of the total. 92.6% and 91.1% of pages in two sets respectively have recall values higher than 0.95, which indicates that our method makes very little useful information lost. We argue that if a page's FM value is larger than 0.8, the page's primary content is located rightly. We achieve 92.6% and 92.4% on two sets respectively. In [15], they achieved 76.9% to locate the most important block of a page on their data set.

Figure 6 illustrates that our algorithm can also achieve good performance on individual sites without any special training. We find that the performance on data set U_3 is excellent, because the site from the domain of gov is of high quality and low cluttered. On data set U_4 we also achieve 90.5% pages with FM value above 0.9.

Two data sets we employ for site-oriented performance evaluation are listed in Table 4, from which 529 pages are gained from four categories of $Sina(www.sina.com)$, and 518 pages from three categories of $QQ(www.qq.com)$.

We select 10 pages from each category to compose training sets, so 40 pages for $Sina$, and 30 pages for QQ. Two site-oriented classifiers(SoC) are obtained as the same as getting UC, then used to their own data set respectively.

Table 3. Universal data sets

Date set	Query/Site	Pages number
U_1	Wanggang	816
U_2	petroleum price	843
U_3	NDRC	524
U_4	Yahoo	502

Table 4. Site-oriented data sets

Data set	Site	Categories	Number
SO_1	Sina	news, finance, sports, mili	529
SO_2	QQ	stock, news, technology	518

Site-oriented method SST mentioned in [17] utilizes the common structure features of the pages that come from the same site, so all the pages from one site must be taken as a whole. The performance of SST and our Mountain Model are compared in Figure 7. Results show that our MM outperforms SST almost in all cases. It indicates that not only can MM be used universally, but also it can get relatively good performance on regular sites with SoC.

Processes of primary content extraction include getting Cell sequence, scoring each Cell and smoothing them, and executing Ridge algorithm. Theoretical analysis of these processes shows that our method's time complexity is linear. Figure 8 shows the time efficiency of our algorithm on data U_1, and we can see the time spent is linear to the number of page processed. About 19.5 pages processed per second, while in [8] the best performance is 3 pages per second. Another advantage of our algorithm is that its memory cost is stable, no matter how many pages processed.

6. Conclusion

To extract primary content from Web pages efficiently and accurately, Mountain Model is proposed to present pages. We use Cell features to eliminate the cluttered links,

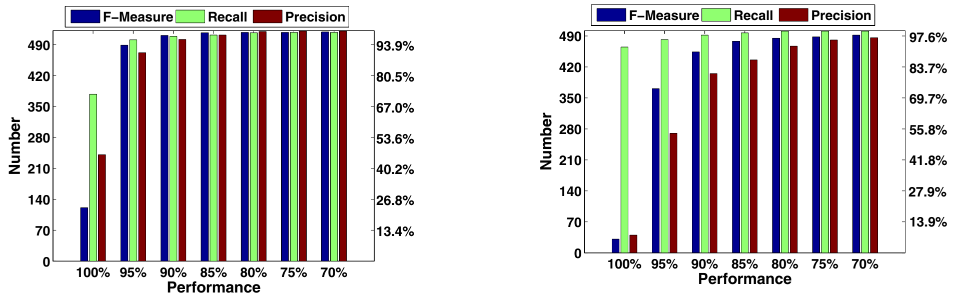

Figure 6. Performances on data sets U_3 and U_4.

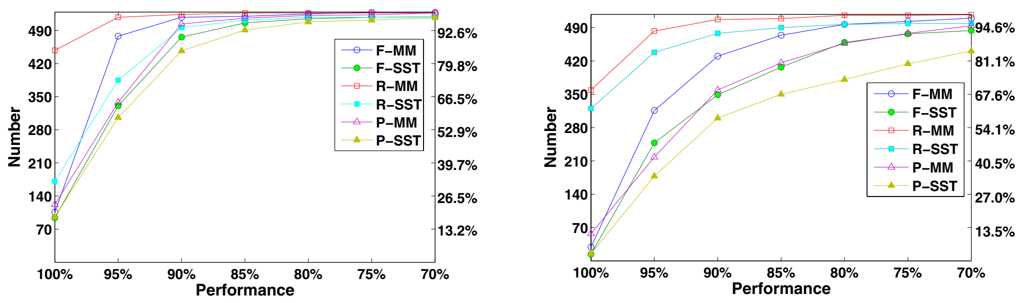

Figure 7. Performances on data sets SO_1 and SO_2.

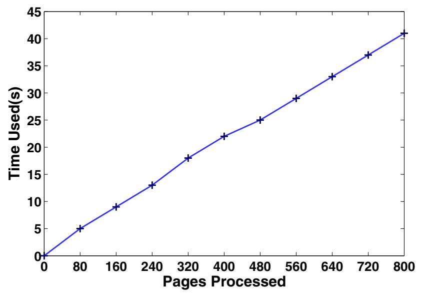

Figure 8. Time efficiency.

and use Ridge features to distinguish primary Ridges from cluttered ones. Our method is site-independent, and experiments show excellent performance is achieved with both accuracy and efficiency compared with existing models.

References

[1] Document object model. http://www.w3.org/DOM/.

[2] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. In *WWW7*, 1998.

[3] D. Cai, X. He, J.-R. Wen, and W.-Y. Ma. Block-level link analysis. In *SIGIR '04*. ACM, 2004.

[4] D. Cai, S. Yu, J.-R. Wen, and W.-Y. Ma. Extracting content structure for web pages based on visual representation. In *APWeb '03*. Springer, 2003.

[5] D. Cai, S. Yu, J.-R. Wen, and W.-Y. Ma. Vips: a vision-based page segmentation algorithm. Technical report, 2003.

[6] D. Chakrabarti, R. Kumar, and K. Punera. Page-level template detection via isotonic smoothing. In *WWW '07*, 2007.

[7] I. Chibane and B.-L. Doan. A web page topic segmentation algorithm based on visual criteria and content layout. In *SIGIR '07*. ACM, 2007.

[8] S. Debnath, P. Mitra, N. Pal, and C. L. Giles. Automatic identification of informative sections of web pages. *IEEE Transactions on Knowledge and Data Engineering*, 2005.

[9] A. Finn, N. Kushmerick, and B. Smyth. Fact or fiction: Content classification for digital libraries. In *Joint DELOS-NSF Workshop*, 2001.

[10] S. Gupta, H. Becker, G. Kaiser, and S. Stolfo. Verifying genre-based clustering approach to content extraction. In *WWW '06*. ACM, 2006.

[11] S. Gupta, G. Kaiser, D. Neistadt, and P. Grimm. Dom-based content extraction of html documents. In *WWW '03*, 2003.

[12] Z. Gyöngyi, H. Garcia-Molina, and J. Pedersen. Combating web spam with trustrank. In *vldb'2004*, 2004.

[13] J. M. Kleinberg. Authoritative sources in a hyperlinked environment. In *SODA '98*, 1998.

[14] S.-H. Lin and J.-M. Ho. Discovering informative content blocks from web documents. In *KDD '02*. ACM, 2002.

[15] R. Song, H. Liu, J.-R. Wen, and W.-Y. Ma. Learning block importance models for web pages. In *WWW '04*, 2004.

[16] W.-C. Wong and A. W.-C. Fu. Finding structure and characteristics of web documents for classification. In *ACM SIGMOD Workshop*, 2000.

[17] L. Yi, B. Liu, and X. Li. Eliminating noisy information in web pages for data mining. In *KDD '03*. ACM, 2003.