Extracting the Main Content from HTML Documents

Samuel Louvan

Technische Universiteit Eindhoven
Den Dolech 2 5612 AZ Eindhoven

Abstract

A modern web document typically consists of many kinds of information. Besides the main content which conveys the primary information, a web document also contains noisy contents such as advertisements, headers, footers, decorations, copyright information, navigation menus etc. The presence of noisy contents may affect the performance of applications such as commercial search engines, web crawlers, and web miners. Therefore, extracting main contents from web document and removing noisy contents is an important process. In this paper we present an approach for extracting main content from web documents which combines classification tasks and heuristic approaches.

1 Introduction

The rapid growth of World Wide Web has been tremendous in recent years. With the large amount of information on the Internet, web pages have boosted the development of information retrieval and data mining applications. However, the web page as the main source of data consists of many parts which are not equally important. Besides the main content, a web page also comprises of noisy parts such as advertisements, headers, footers, that can degrade the performance of information retrieval applications. Therefore, an approach to identify and extract main content is needed to alleviate this problem.

There are numerous approaches to perform web content extraction. Typically, the extraction involves web page segmentation and the use of heuristics rule set to locate the main content. Different approaches have been proposed for web segmentation algorithm and can be divided into three categories namely DOM-based [1, 2, 3], Vision-Based [4] and densitometry approaches [5].

One of the tools that is similar to our work is Crunch [2, 3]. It uses DOM based approach to perform web page segmentation and a set of heuristic rules to discover the main content. Crunch includes advertisement remover by maintaining a list of advertisement server, an empty table remover etc. The approach in [4] uses VIPS algorithm to do web page segmentation and classifies segments into three levels of importance. Many other approaches to determine main content exist such as Largest Pagelet [8], Link Quota Filter [2], Text to Tag Ratio [7], Document Slope Curve [6]. The latest developed approach such as Content Code Blurring (CCB) and its variants [18] use the image blurring paradigm in order to detect the main content. Most of the approaches are heuristic based, created based on human observation and focused on certain feature.

2 Our Approach

Generally, most of the existing content extraction approaches used heuristics rules in order to achieve their goal. Typically, for heuristic rules they introduced one main feature to distinguish main content from noisy information. However, most of the time, by introducing only a single main feature will not solve the problem. Combination of different features usually can lead to a better content extraction as different kinds of web pages have different characteristics.

In addition to that, the existing approaches, except Crunch, basically treat all the web pages the same way. For example, there will not be any difference between extracting main content from news web pages, blog web pages, and web forums.

The approach that we take is based on the combination of machine learning and heuristic approach. There are several reasons for choosing this approach. First, by using the learning process, we can combine many kinds of different features and the learning algorithms may deduce the parameters for the features automatically. In this way, at least it will reduce the amount of work to specify appropriate feature parameter values. Second, by using this approach we may train the algorithm to extract content from different types of web pages. For example, we can build a classifier that specializes on news web pages, web blog pages, web forum pages etc. As an overview, our approach can bee seen as a one pipeline process.

As shown as in Figure 1, the input of our approach is the HTML document and the output is the text strings that are extracted from the HTML document. In between, there are two processes namely classification and heuristic rules. The classification process is used to identify the segments which contain the main content. The output of the classification task is one or more segments (DOM nodes) which classified as main content. These segments will be the input of the heuristic rules which performs further filtering to extract the text string of the main content. The following sections will explain about the classification and heuristic processes.

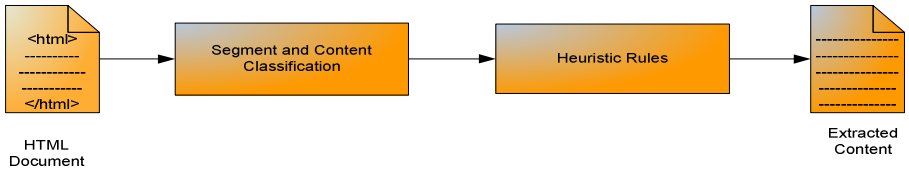

Figure 1 Overall Approach

2.1 Segment Classification

The purpose of segment classification is to break down the structure of a web document into smaller segments with certain granularity. We based our segmentation process on the Document Object Model (DOM)[1] of the web document. According to the HTML 4.01 specification [14], there are several HTML elements which define structural element such as block level elements (`DIV`, `SPAN`), list elements (`UL`, `OL`, `LI`), table elements (`TABLE`, `TR`, `TD`).

We traverse the nodes in the DOM tree in a depth first manner and for each node that we encounter; we check whether a node is a structural elements. If it is a structural element then the classifier will categorize whether it is suitable to be a good segment or not. The meaning of *suitable* in this context is related to the granularity of the segments. A segment could be very coarse or fine-grained. In general, we do not prefer segments which are extremely coarse or fine grained.

What we prefer as a segment with good granularity is a segment which represents uniform semantic unit. For human, it is very easy to distinguish the semantic of segments in a web document. In the other way around, it is difficult for machine to do this. In order to select the right level of granularity we used supervised learning to train the algorithm to recognize a segment with a good granularity. We view this problem as a classification task whether a DOM node is a good segment or not. One of the heuristic that we used to control the granularity is the DOM height which is the maximum depth that can be reached from a particular DOM node to a certain leaf node.

2.2 Content Classification

The outputs of segment classification are the DOM nodes which are classified as good segments. These segments will be used for content classification process. The content classification basically takes the feature vector of a segment and then classifies it into a main content or noisy content.

[1] According to W3C specification, DOM is an application programming interface (API) for valid HTML and well-formed XML documents.

The representation of our training data is represented as a feature vector:

$$< X_1, X_2, X_3 \ldots X_N, Y >$$

Where each X_i is a feature and Y is the class label. The features used here are the properties that we can obtain from DOM properties and visual properties.

The features that we use are as follows:

- Features from DOM properties: inner text, inner html, number of images, number of forms, number of table elements, number of links etc.
- Visual representation: position coordinates of the DOM node, width and height.
- Number of stop words inside the DOM node.
- Ratio between anchored text and the inner text.
- DOM Height: the maximum depth that can be reached a particular DOM node to a certain leaf node.

2.3 Additional Heuristic Approaches

After the classification process is conducted, the returned DOM nodes may have problems. The example of these problems e.g. in many cases there are embedded noisy contents such as embedded advertisements, link to related articles, short snippets of the article etc. Comments from the web page visitors also may jeopardize the classification result. Therefore, it is not a wise decision to extract the content by only traversing all the text contents in the DOM nodes and get the text strings. In order to alleviate this problem we apply heuristic approaches.

2.3.1 Largest Block String (LBS)

Based on our observation, there is a common pattern of text nodes that contain main content in a news or blog page. The text strings are always placed contiguously in the same depth level of a DOM sub tree. Most likely, the text strings are placed inside a `P` tag but however it also can be placed inside other tags such as `DIV` or plain `#text` node.

The way we find the largest block of string is by traversing the DOM tree depth first based and then for each level in the sub-tree we find a child node with the longest string. After that we look to the sibling nodes of the child node in order to discover the neighboring text strings. We will add the sibling's text string if it contains `#text` node. In order to skip the embedded noisy information such as advertisements or scripts we check the content of the sibling's node. If it doesn't contain `#text` string as a child, we will skip it. We repeat this process until all the sub trees are examined. In the end, the output is the largest block of text string found in a sub tree.

2.3.2 String Length Smoothing (SLS)

The string length smoothing is inspired by the approach of [7] and [11]. In [7], they used HTML tag to text ratio and then perform smoothing function and clustering. The difference with [11] is that they use the windowing and cutoff factor relative to maximum string length to decide whether a string should be regarded as content. In this heuristic we tried to experiment with the length of string inside a DOM node which is classified as a main content then perform smoothing. SLS work as follows; we traverse into all the text nodes and put the string's length into a one dimensional array. Every time we encounter the HTML tags that modify the structure such as `DIV, TR, TD, TABLE, UL, LI`, we put an empty element into the array.

Figure 2 shows the length of value of the string inside a DOM node in one of the web page in our data set. It can be seen from the figure that in the middle of the chart, there are several strings with large length. Likewise, the neighboring strings of the large string there are also small strings which are actually part of the main content. In order to detect these small strings, we perform the value smoothing to spread the value of a string to the neighboring strings.

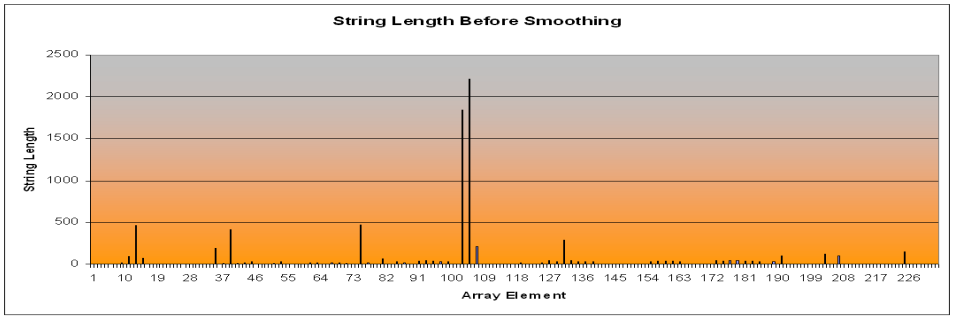

Figure 2 String Length Before Smoothing

We used the formula in [7] to calculate the smoothing value. The formula is shown in Equation 1. We smooth the value in *StringArray* by computing the new value e_k for a given radius (r) for each element (k). The resulting smoothing values are shown in Figure 3. It can be seen that there are increases of string length for the small strings which are located near the large strings.

$$e_k = \frac{\sum_{i=k-r}^{k+r} StringArray[i]}{2r+1}$$

Equation 1 Smoothing Function

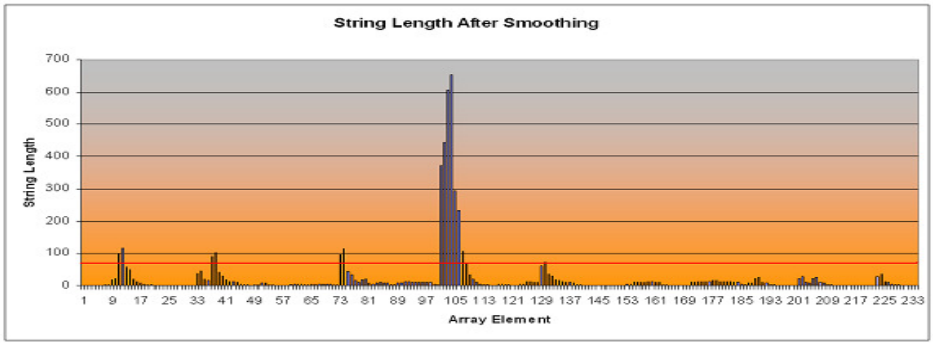

Figure 3 String Length After Smoothing With Standard Deviation of 73.85 (red line)

After we performed the smoothing, the next task is to choose which strings to extract. In order to do this we used a threshold value. The standard deviation of the string length is set as a threshold. We extract the strings which have the length equal or larger than the standard deviation.

2.3.3 Table Pattern (TP)

The table pattern (TP) is developed mainly to extract the content from web forums. The heuristic comes from the observation that almost all web forums are presented in a table like structure. Among forum posts in the same page, they share the same pattern of the table.

Moreover, we can observe this similarity from the internal DOM structure. The tables which contain the forum posts share almost the same properties such as CSS Class, border, width, align, cell spacing, cell padding etc.

3 Experimental Setup

3.1 Data

For our experiment we chose three different domains namely news, blogs and forums. Some of the news dataset were obtained from Gottron [18]. Meanwhile, for forums and blogs dataset we decided to make it our own. The total pages among the datasets are shown in Table 1.

Table 1 Experiment Dataset

Dataset Type	Number of Sources	Total Pages
News	14	1750
Forum	4	300
Blog	80	1070

3.2 Training

Our approach requires labeled training data; therefore we developed a web based tool so that users can label the area of the web page as a main content or a noisy content, a good segment or a bad segment. For each dataset type we take 10% of the pages as a training set and the rest as an evaluation set.

The pages in the training set are taken from different sources which do not occur in the evaluation set e.g. if the evaluation set consist of Yahoo news pages, there will not be any Yahoo news pages in the training set. In addition to that, we need to label the exact text string that is expected to be extracted as a main content. We called this text string as a gold standard. In order to establish the gold standard, we wrote a static content extractor (written in C#) for each website source to extract the content.

4 Results

4.1 Classification Results

In order to select the best learning algorithm for the classification process, we performed 10-fold cross validation to our training datasets. We experimented with several classifiers from WEKA and used default parameter settings. The results that we obtained for both segment and content classification in terms of precision, recall, and F1 measure are reasonably high. Regarding the number of training instances, the distribution are as follows, 1356 instances for forum dataset, 2030 instances for blogs dataset, and 2030 instances for news dataset.

Table 2 Segment Classification Result

Learning Algorithm	News			Blogs			Forum		
	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
Decision Tree (J48)	0.964	0.962	0.963	0.952	0.975	0.963	0.988	0.988	0.988
Random Forest	0.966	0.976	0.971	0.963	0.988	0.975	0.992	0.993	0.992
SMO	0.906	0.902	0.904	0.907	0.99	0.947	0.926	0.99	0.957
Multilayer Perceptron	0.942	0.969	0.955	0.948	0.966	0.957	0.988	0.983	0.986

Most of the classifiers can achieve more than 90 % score for all measurements as shown as in Table 2 and Table 3. In general, for all dataset types, two classifiers namely random forest and J48 produce the best result. Motivated by this encouraging result, we integrated the decision tree classifier in our content extraction tool.

Table 3 Content Classification Result

Learning Algorithm	News			Blogs			Forum		
	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
Decision Tree (J48)	0.984	0.961	0.972	0.922	0.94	0.931	0.958	0.98	0.969
Random Forest	1	0.969	0.984	0.939	0.97	0.954	0.973	0.98	0.993
SMO	0.939	0.969	0.953	0.829	0.95	0.885	0.749	0.921	0.826
Multilayer Perceptron	0.992	0.984	0.988	0.929	0.938	0.934	0.95	0.98	0.965

After we performed the cross validation, we obtained the best classifier model for each dataset type. After that, for the content extraction process we applied appropriate classifier model to a dataset type e.g. the news classifier model to the news dataset type, the blog classifier model to the blog dataset type etc.

4.2 Content Extraction Result

As the final evaluation, we need to measure the performance of the content extraction. For every web page in the evaluation data set we compare the gold standard and the extracted content. In order to measure the similarity of the two strings we used the Longest Common Subsequence (LCS) [19] of the strings.

Likewise, as an evaluation metric we use precision, recall, and F1 measure. In order to apply these measures we need to find the overlap between the gold standard and the extracted content. Looking at the LCS as the intersection of the gold standard and the extracted content, the computation of recall (*r)*, precision (*p),* and F1 measures are defined in Equation 2.

$$p = \frac{|c|}{|g|} \quad r = \frac{|c|}{|e|} \quad F1 = \frac{2 \times p \times r}{p + r}$$

Equation 2 Precision, Recall, and F1 of LCS

|c| is the string length of the LCS of the two strings. |g| is the length of the gold standard string. |e| is the length of the extracted content string.

As we mentioned before, we evaluated three kinds of dataset namely news, blogs, and forums. Actually, in [18] Thomas Gottron compared many kinds of approaches but here we only presented several of them which are the nearest competitor to our approach. We called our approach as *content-seeker (CS)* and in the evaluation we also combine the machine learning technique with our heuristic approach namely LBS and TP.

Table 4 Average precision, recall, F1, and standard deviation for news dataset evaluation

Method	Precision	Recall	F1
CS+LBS	**0.940±0.183**	0.882±0.187	**0.905±0.182**
TCCB-25	0.806±0.228	0.859±0.198	0.816±0.197
CCB-r40	0.796±0.286	0.753±0.285	0.763±0.273
DSC	0.863±0.167	0.870±0.136	0.852±0.141
Crunch	0.65±0.260	**0.964**	0.714

Table 5 Average precision, recall, F1, and standard deviation for blog dataset evaluation

Method	Precision	Recall	F1
CS+LBS	**0.905±0.223**	0.827±0.293	**0.834±0.279**
TCCB-25	0.566±0.347	0.907±0.200	0.634±0.307
CCB-r40	0.593±0.352	0.878±0.234	0.648±0.311
LQF-75	0.268±0.195	**0.983±0.109**	0.386±0.220

Table 6 Average precision, recall, F1, and standard deviation for forum dataset evaluation

Method	Precision	Recall	F1
CS+TP	**0.870±0.183**	0.740±0.326	**0.754±0.279**
LP-7	0.428±0.283	0.554±0.302	0.414±0.245
LQF-75	0.296±0.196	**0.974±0.154**	0.417±0.223

According to our web content extraction evaluation results in Table 4, 5, and 6, for the news dataset and blog dataset, CS+LBS is the best approach. In terms of precision, CS+LBS shows that it always yield the best score in both datasets. In terms of recall, CS+LBS is outperformed by LQF, Crunch. However, these methods yield very low precision score. Therefore, the CS+LBS is preferable since it is best in average precision and still gives reasonably high score of average recall thus it obtains the best F1 score.

For the forum dataset, according to our evaluation, the CS+TP is the best approach. The CS+TP yields the best performance in precision however it has tradeoff in terms of recall. The LQF variants are the best in recall, nevertheless its precision scores are very low. Therefore, we claim that CS+TP is the best approach for the forum dataset.

5 Conclusion

In general, according to our experiments we have shown that our approach namely the combination of machine learning (classification) and our own developed heuristic approach is competitive compared to the existing heuristic methods. We introduced several heuristics that can be used in the news, blogs and forums datasets namely Largest Block of String (LBS), String Length Smoothing (SLS) and Table Pattern (TP).

Essentially, our hybrid approach manages to perform better than the existing related works especially in terms of precision. As an overall result, the hybrid approach as a combination of LBS and classification tasks namely CS+LBS yields the best performance for blogs and news datasets. LBS based on the assumption that the main content usually is placed in a large contiguous text node which is suitable for news and blogs web pages.

As for forum dataset, we developed the other heuristic namely TP. TP is based on the observation that forum web pages always presented in a table-like structure. According to our experiment, the combination of classification and TP, CS+TP gave reasonably high score of precision and F1 compared to the other existing heuristic methods

6 Acknowledgement

We want to thank Mykola Pechenizkiy for his valuable inputs. We also want to thank TU Eindhoven /ASML Scholarship Foundation and Teezir B.V. who have supported this project. Thomas Gottron for the discussion regarding the evaluation method and his willingness to share his dataset.

References

[1] Suhit Gupta, Gail Kaiser, David Neistadt, and Peter Grimm. *DOM-based content extraction of HTML documents*. In WWW '03: Proceedings of the 12th International Conference on World Wide Web, pages 207–214, New York, NY, USA, 2003. ACM Press.

[2] Suhit Gupta, Gail Kaiser, and Salvatore Stolfo. *Extracting context to improve accuracy for HTML content extraction*. In WWW '05: Special Interest Tracks and Posters of the 14th International conference on World Wide Web, pages 1114–1115, New York, NY, USA, 2005. ACM Press.

[3] Suhit Gupta, Gail E. Kaiser, Peter Grimm, Michael F. Chiang, and Justin Starren. *Automating Content Extraction of HTML Documents*. World Wide Web, 8(2):179–224, 2005

[4] Song, R. Liu, H., Wen, J.-R, W.-Y, Ma. *Learning Block Importance Models for Web Pages*. WWW-04, 2004.

[5] Christian Kohlschutter, Wolfgang Nejdl. *A Densitometric Approach to Web Page Segmentation.* CIKM'08.

[6] David Pinto, Michael Branstein, Ryan Coleman, W. Bruce Croft, Matthew King, Wei Li, and Xing Wei. *QuASM: a system for question answering using semi structured data*. JCDL'02: Proceedings of the 2nd ACM/IEEE-CS Joint Conference on Digital Libraries.

[7] Tim Weninger and William H. Hsu. *Text extraction from the web via text-tag- ratio*. In TIR '08: Proceedings of the 5th International Workshop on Text Information Retrieval, pages 23 – 28. IEEE Computer Society, September 2008.

[8] Ziv Bar-Yossef, Sridhar Rajagopalan. *Template Detection via Data Mining and its Application.* ACM WWW Conference 2002

[9] Sandip Debnath, Prasenjit Mitra, C. Lee Gilles. *Automatic extraction of informative blocks from web pages*. SAC'05 Proceedings of the 2005 ACM Symposium on Applied Computing, pages 1722-1726, New York, NY, ACM Press. 2005

[10] Shian-Hua Lin, Jan-Ming Ho, *Discovering Informative Content Blocks from Web Documents*. ACM SIGKDD. 2002

[11] Javier Arias Moreno, Koen Deschact, Marie-Francine Moens. *Language Independent Content Extraction from Web Pages.* Proceedings of the 9th Dutch - Belgian Information Retrieval Workshop Page 50-55. 2009.

[12] Bing Liu, Kevin Chen-Chuan Cang. *Editorial: Special Issue on Web Content Mining*. SIGKDD Explorations. Volume 6, Issue 2.

[13] http://www.w3.org/TR/2000/WD-DOM-Level-1-20000929/introduction.html

[14] http://www.w3.org/TR/html401/struct/global.html

[15] http://www.cs.waikato.ac.nz/ml/weka/

[16] http://www.informatik.uni-mainz.de/~gotti/publications.php

[17] http://technorati.com/pop/blogs/

[18] Thomas Gottron. *Content Extraction: Identifying The Main Content in HTML Documents*. Dissertation. 2008. http://ubm.opus.hbz-nrw.de/volltexte/2009/1859/pdf/diss.pdf

[19] D.S. Hirschberg. *A Linear Space Algorithm for Computing Maximal Common Subsequence*. Communications of ACM Volume 18 Number 6, 1975, p.342.