Noise Removal Methods from Web Pages

Vasilii Korelin
Saint-Petersburg State University
7-9 Universitetskaya Naberezhnaya,
St. Petersburg, 199034, Russia
+7 911 266 12 99
vn.korelin@gmail.com

Ivan Blekanov
Saint-Petersburg State University
7-9 Universitetskaya Naberezhnaya,
St. Petersburg, 199034, Russia
+7 921 339 53 43
i.blekanov@gmail.com

ABSTRACT

Almost all the pages of websites of large organizations have a variety of markups, headers, footers and menu items. In terms of information retrieval, this part of the page is not semantically significant for it and can be considered as noise. Furthermore, noise can negatively affect information retrieval results. Therefore, eliminating noisy information is an important step in pre-processing for subsequent analysis (clustering, classification, etc.). This paper discusses several methods of noise removal from Web pages belonging to a large collection. The first method is based on the use of Boilerpipe library to detect and remove surplus "clutter" (boilerplate, templates) around the main textual content of a Web page. The second method is based on a headless browser. The third method involves the use of HTML5 semantic markup (only applicable for browsers that support HTML5). An experiment to assess the quality and performance speed of the methods described is presented. Comparative analysis is carried out.

Categories and Subject Descriptors

H.2.8 [**Database Applications**]: Data mining; H.3.3 [**Information Search and Retrieval**]: Information filtering.

General Terms

Algorithms, Performance, Experimentation, Languages.

Keywords

Noise detection, noise elimination, HTML5, Boilerpipe, headless browser, document preprocessing.

1. INTRODUCTION

For today, in connection with the large number of websites it became necessary to analyze content of Web pages for further information processing (information retrieval, classification, clustering) [1]. Humans can easily distinguish the main page content from various noisy information such as navigation menus, header and footer elements, advertising and other text portions during website examining. This main content can be considered as semantically significant information. Normally, the Web crawlers that gather Web pages eliminate noise extract semantically significant information and store it for further analysis. There is a variety of techniques for noise detection in Web pages [2]. In this paper we consider three approaches of noisy information removal that are based on the following tools: Boilerpipe library, HTML5 semantic markup, Headless browser with Selenium.

This paper is structured as follows. The Section 2 describes methods that can be used for extraction of main textual content. An experiment to assess the quality of the methods described is presented in the Section 3. The Section 4 is devoted to conclusion.

2. NOISE REMOVAL TOOLS

2.1 Boilerpipe library

The Boilerpipe library was written by Christian Kohlschutter for Java platform and released under the Apache License 2.0. It is based on the algorithms that provide detection and removal the surplus "clutter" (templates, boilerplate) which is placed around the main textual content of a Web page. Nowadays, the library supplies specific approaches for common task (e.g., extraction of the news articles) and can be easily enlarged for personal problem issues. The paper "Boilerplate Detection using Shallow Text Features" was the basis of the Boilerpipe library that uses and extends algorithms from it [3]. Analysis of small set of shallow text features is the main approach for classifying the particular text elements of a Web page. The main components of the Boilerpipe library are listed below.

HTML parser

The main purpose of the HTML parser is to transform an HTML page into set of text "blocks" that can be considered as an internal text-only document model. The parser is based on the third-party library, CyberNeko [4]. It transforms an HTML document into a TextDocument which consists of one or more TextBlocks. HTML parser can distinguish specific elements such as Script, Option, etc. and ignore them automatically. Each TextBlock contains a text portion from the HTML document and shallow text statistics for it (e.g., words number and words number in the anchor text).

Filters

Every filter is applied to the TextDocument and normally iterates once over all TextBlocks from it. Every individual TextBlock is marked by filter as content or boilerplate. Also filter can assign additional textual label to it. Boilerpipe's filters are grouped as follows:

1. Basic filters

2. English filters, that can be applied to English text (they might also be applied to the other Western languages, but some settings perhaps need to be changed).
3. Heuristics Filters was not explored but will be investigated in the future.

Extractors

Extractors are formed by one or more Filters. Their main purpose is to obtain content from a Web page. For instance, extractor that uses "pipelines" filters takes the parsed HTML document and extracts the main textual content from it. There are several different extractors, from generic DefaultExtractor to special extractors (e.g., ArticleExtractor which is used for news articles).

HTML highlighter

HTML highlighter is an additional tool to represent extracted main content of a Web page as an HTML document.

The algorithms used in Boilerpipe are quite content-independent. However, if the page contains insufficient HTML text (i.e., PDF, Flash or JavaScript) the correctness of the library work is not guaranteed.

2.2 HTML5 Semantic Markup

Such tags as <div> and <span> have little meaning for those who examines an HTML code whether it is machine or human. <div> elements are typically designed for positioning the content on the page. <span> elements are responsible for special formatting of the content. Developers have been using <div> elements to combine page layout, and the developer usually provides the meaning of each <div> element which is based on its id or CSS class.

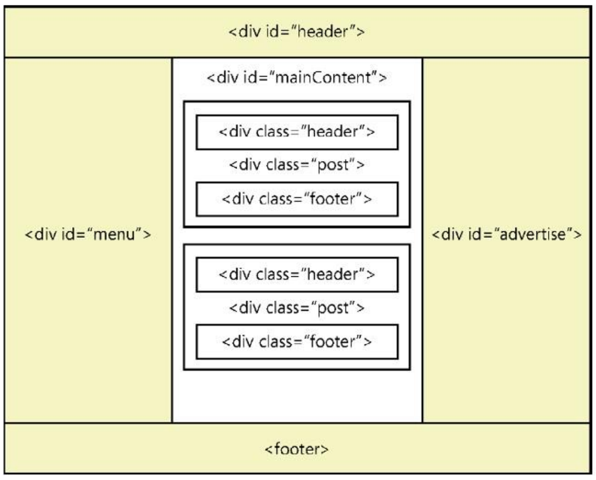

Figure 1. A blog site layout container using <div> elements

HTML5 separates presentation, structure and behavior. Semantic is defined as the study of meaning of linguistic expressions [5]. The HTML5 standard introduces tags that do not serve for presentation purposes but provide meaning. Users usually do not read an HTML code during website surfing, but many machines read it to interpret the Web page. Moreover, Nonvisual Desktop Access (NVDA) devices can provide other means of Web pages processing.

HTML5 semantic elements can be used for creation of layout container which has elements that are meaningful to both the developer and the machine. The following are common elements by which an HTML5 layout container can be created.

- <header> specifies a header section of HTML document and can be used as a page header. Moreover, it can be place at the <article> element.
- <footer> specifies a section that is placed at the bottom of the HTML document. Moreover, it can be placed at the bottom of the <article> tag content.
- <nav> specifies a section that contains a block of links used for navigational purposes.
- <aside> specifies a section that is normally used for sidebars.
- <section> specifies a section that contains <h1> to <h6> internal elements.
- <article> specifies a section that can be considered as a unit of content (e.g., blog posts, news articles).

In Figure 2, all <div> elements have been replaced with the HTML5 semantic elements.

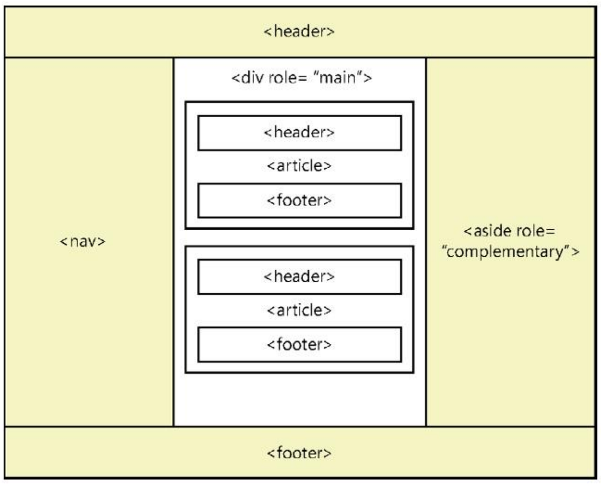

Figure 2. Layout container example, using the new HTML5 elements

Web Accessible Initiative (WAI) specifies the Accessible Rich Internet Applications (ARIA) suite (WAI-ARIA). WAI-ARIA defines a role classes that can used for providing additional meaning for HTML page elements. For instance, screen readers can use it for accessibility purposes [6]. There are several parent and child role classes. "Landmark" class is a parent role class that describes regions of the Web page and can be set as an attribute of the HTML tag. The following are child classes of the "Landmark" role class.

- *Application* defines an area that is declared as a Web application.

- *Banner* defines an area with site-specific content (e.g., site name, logo).
- *Complementary* defines an area for an additional page content that can have different meaning than main textual content.
- *Contentinfo* defines an area that contains information about document (copyright notices, links). It is typically footer content, one per Web page.
- *Form* defines an area that contains input controls sending gathered data to server.
- *Main* defines an area for the main content of Web page.
- *Navigation* defines an area for navigational links.
- *Search* defines an area of input controls for query entering and displaying corresponding information.

An <article> tag can be used for noise removal from Web pages. Furthermore, text that is placed into tags with Role attribute which have an appropriate value (i.e., Main) can be also considered as semantically significant.

2.3 Headless browser

One of the methods of Web site content extraction is based on Headless browser that is a Web browser without a graphical user interface. Usage of the headless browser provides a control a Web page via a command line interface or other network communications in environment that is similar to popular browsers. Using Selenium framework that is designed for testing of Web applications user has access to various elements on the Web page [7]. Today, Selenium does not provide methods of noisy information removal. Without any additional methods Selenium framework can extract only all Web page textual content. Therefore the amount of noisy information can be determined among the all textual content of web page. Also Jaccard similarity for headless browser can be compared with Jaccard similarity for other noise reduction methods and determine if these methods should be used for extraction of semantically important text. For instance, method is not appropriate for extraction semantically significant information in case of Jaccard similarity evaluated for this method is less than Jaccard similarity evaluated for headless browser.

3. METHODS EVALUATION

For comparative analysis of methods considered in Section 2 test documents collection was chosen which contains 30 documents with different design, topics and amount of semantically significant information. In these documents all the noise was removed by expertise (manually) and the main content was extracted. This content can be considered as the expected text in comparison with the text that is obtained using the methods of noise removal. In the current paper Jaccard similarity and Shingles algorithm are used to evaluate the similarity of two texts [8].

3.1 Jaccard similarity

A document represents a string of characters. Shingling is a process that creates sets of k-shingles. Define a k-shingle for a document to be any substring of length k found within the document. For instance, $k = 2$, doc="a b c a b". Then after shingling next sets can be obtained: {a, b}, {b, c}, {c, a}. Similar documents to each other will have a lot of equal shingles [8].

In this paper Jaccard similarity is used to determine the similarity of two documents. The Jaccard similarity of sets C_1 and C_2 is the ratio of the cardinality of the intersection of C_1 and C_2 to the cardinality of their union [9].

$$Sim(C_1, C_2) = \frac{|C_1 \cap C_2|}{|C_1 \cup C_2|}$$

Every document can be considered as a set of shingles where every shingle contains k words. Therefore two documents can be represented as boolean matrix with two columns. Rows correspond to the elements of the universal set which is the set of all k-shingles. Columns correspond to the documents. Value of 1 is in the row E and in the column S if and only if the document that corresponds the column S contains the single that corresponds the row E. Otherwise, it is value of 0. Therefore, documents similarity is the Jaccard similarity of the sets of their shingles.

$$\begin{array}{c|cc} & \underline{C_1} & \underline{C_2} \\ \hline a & 1 & 1 \\ b & 1 & 0 \\ c & 0 & 1 \\ d & 0 & 0 \end{array}$$

Figure 3. Boolean matrix representing two documents

$Sim(C_1, C_2) = \frac{1}{3}$ – similarity of two documents.

3.2 Experimental results

Tables 1, 2, and 3 present the Jaccard similarity between the expected main textual content and the text that was extracted by the methods discussed above. Shingles of different lengths (2, 5, 10 and 15 words per shingle) are used. The test set of documents was divided into 3 groups: sites with small (Table 1), middle (Table 2) and large (Table 3) main textual content.

Table 1. Comparison of the noise removal results for small size main textual content

Method	k=2	k=5	k=10	k=15	Average
Boilerpipe library	73%	71%	69%	68%	70.25%
HTML5 Semantic markup	49%	48%	48%	47%	48%
Headless browser + Selenium	24%	23%	22%	21%	22.5%

Table 2. Comparison of the noise removal results for middle size main textual content

Method	k=2	k=5	k=10	k=15	Average
Boilerpipe library	83%	82%	80%	78%	80.75%
HTML5 Semantic markup	78%	76%	73%	71%	74.5%
Headless browser + Selenium	71%	70%	69%	68%	69.5%

Table 3. Comparison of the noise removal results for large size main textual content

Method	k=2	k=5	k=10	k=15	Average
Boilerpipe library	93%	88%	83%	79%	85.75%
HTML5 Semantic markup	87%	83%	78%	75%	80.75%
Headless browser + Selenium	79%	75%	71%	68%	73.25%

Comparative analysis shows that the more semantically significant information is present on the Web page the more Jaccard similarity between the expected main textual content and the text obtained by described methods as the more textual content is on the Web page, the more shingles are in the intersection for both documents. Moreover Jaccard similarity grows as the number of words per shingle (k) decreases because for small values of k there are more shingles in the intersection than for large k value. For large size main textual content and small k value Jaccard similarity is greatest because in large size text the shingle with 2 words is much more common for two documents than the shingle that contains 10 or 15 words. For documents with small and middle size textual content the number of shingles in intersection for $k = 2$ is not much different from number of shingles in intersection for $k = 15$. This behavior is typical for all methods.

The highest Jaccard similarity coefficient was shown by the method that is based on Boilerpipe library. Usage of the HTML5 semantic markup eliminates noisy information with less accuracy and can be applied only for those Web sites that use HTML5 semantic markup. Headless browser with Selenium framework renders the entire Web page but cannot remove noisy information. As expected, Jaccard similarity of the last method is significantly lower than Jaccard similarity of the other methods.

4. CONCLUSIONS

This paper describes three methods of noisy text removal for extraction of semantically significant information from Web pages. These methods were tested on a collection of Web pages that have varying markup and amount of main textual content. Experiment shows that Boilerpipe library removes noisy information with the highest accuracy for all kinds of Web pages.

5. ACKNOWLEDGMENTS

This work was supported by the Russian Foundation for Basic Research, grant № 15-01-06105.

6. REFERENCES

[1] Chakrabarti, S. Mining the Web: Discovering Knowledge from Hypertext Data. Morgan Kaufmann, 2002.

[2] Lan Yi, Bing Liu and Xiaoli Li. Eliminating noisy information in Web pages for data mining. Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining, August 24-27, 2003, Washington, D.C.

[3] Christian Kohlschütter, Peter Fankhauser, Wolfgang Nejdl. Boilerplate detection using shallow text features. WSDM'10: Proceedings of the third ACM international conference on Web search and data mining, January 2010.

[4] A. Clark and M. Guillemot. CyberNeko HTML Parser. http://nekohtml.sourceforge.net/

[5] Glenn Johnson. Programming in HTML5 with JavaScript and CSS3. Microsoft Press, A Division of Microsoft Corporation, One Microsoft Way. Redmond, Washington 2013. 205–212

[6] Peter Thiessen, WAI-ARIA live regions and HTML5, Proceedings of the International Cross-Disciplinary Conference on Web Accessibility, March 28-29, 2011, Hyderabad, Andhra Pradesh, India.

[7] Alexander Sirotkin. Web application testing with selenium. Linux Journal, v.2010 n.192, p.3, April 2010.

[8] Vasilii Korelin, Ivan Blekanov, Sergey Sergeev. Clustering of the External WEB Environment of Universities Using a Modified LSH Algorithm. St. Petersburg State Polytechnical University Journal. Computer Science. Telecommunication and Control Systems. Vol. 5, 2015. 79-87.

[9] Jatsada Singthongchai and Suphakit Niwattanakul, "A Method for Measuring Keywords Similarity by Applying Jaccard’s, N-Gram and Vector Space," Lecture Notes on Information Theory, Vol.1, No.4, pp. 159-164, Dec. 2013. doi: 10.12720/lnit.1.4.159-164.