International Journal of Computer Science and Engineering (IJCSE)
ISSN(P): 2278-9960; ISSN(E): 2278-9979
Vol. 3, Issue 2, Mar 2014, 27-34

International Academy of Science, Engineering and Technology
Connecting Researchers; Nurturing Innovations

CONTENT EXTRACTION USING DOCUMENT OBJECT MODEL AND NATURAL LANGUAGE PROCESSING FOR WEB

AAKRITI AGARWAL, SHAILEY CHHEDA, KRIMA SHAH & MEERA NARVEKAR

Department of Computer Engineering, Mumbai University, DJSCOE, Mumbai, Maharashtra, India

ABSTRACT

Web pages often contain clutter such as pop-up advertisements, unnecessary images and extraneous links around the body of an article that distract a user from actual content and may reduce effects of many advanced web applications. Often this noisy content is combined with the main content leaving no clean boundaries between them. This noisy content as a result makes the problem of information harvesting from web pages much harder. Most approaches to removing clutter or making content more readable involve changing font size or removing HTML which takes away from a webpage its inherent look. Unlike 'Content Reformatting', which aims to recreate the webpage in a more convenient form, our solution directly addresses 'Content Extraction', an approach that does not require previous knowledge of website templates. For higher accuracy in content extraction, the analyzing software needs to act as a human user and understand content in natural language along with HTML DOM analysis in order to eliminate noisy content. In this paper, a combination of HTML DOM analysis and Natural Language Processing (NLP) techniques for automated extractions of main article of interest with associated images from web pages has been described**.**

KEYWORDS: Content Extraction, DOM, NLP

INTRODUCTION

Considering that a huge amount of world's information resides in web pages, it is becoming very important to analyze and mine information from web pages. HTML format is designed more for human users to view and does not easily lend itself to automated processing for information extraction. A seemingly simple visual block containing a few paragraphs of text is typically coded using tens of HTML nodes some of which contain the text and others contain styling information to control layout. This problem is compounded by the fact that typical web pages contain significant amount of unrelated content interspersed with the main content. The noisy content is mostly in the form of navigation bars on top and/or on the side, horizontal or vertical banner advertisements, boxes with links to unrelated content, boxes containing images or animated advertisements, etc. Such noisy content adversely affects performance of web content analysis and information extraction technologies.

Thus, it is essential to extract the main article content including all the text with original styling information along with any associated images or figures with their captions while eliminating other images which are advertisements and other unrelated text blocks. Content Extraction can be used to remove clutter without destroying webpage layout, making more of a page's content that can be viewed at once.

Content extraction is particularly useful for the visually impaired and blind. A common practice for improving web page accessibility for the visually impaired is to increase font size and decrease screen resolution; but, this also increases the size of the clutter, and thus reducing effectiveness. Screen readers for the blind, like 'Hal Screen Reader' by

Dolphin Computer Access or Microsoft's Narrator, don't usually automatically remove such clutter either and often read out full raw HTML. Thus, both groups benefit from extraction as less material must be read to obtain the desired results.

A generic solution employs a series of techniques that address the aforementioned problems. In order to analyze a web page for content extraction, we pass web pages through an HTML parser that corrects the markup and creates a Document Object Model tree. The Document Object Model is a standard for creating and manipulating in-memory representations of HTML (and XML) content. Increasing support for the Document Object Model makes the solution widely portable.

Natural Language Processing (NLP) and information retrieval (IR) algorithms can also benefit from content extraction, as they depend on the relevance of content and the reduction of standard word error rate to produce accurate results. Content extraction allows the algorithms to process only the extracted content as input as opposed to cluttered data coming directly from the web. Currently, most NLP-based information retrieval applications require writing specialized extractors for each web domain. While generalized content extraction is less accurate than hand-tailored extractors, they are often sufficient and reduce labor involved in adopting information retrieval systems.

In this paper, we first describe related work and applications. Then we give a detailed problem statement and finally we describe various approaches to solve the problem.

RELATED WORK

The term Content Extraction (CE) was introduced by Rahman et al. [1] in 2001. In the last decade, extraction of content from web pages has been studied intensively and numerous methods have been developed. There are sets of content extraction approaches based on statistical information of web pages. In 2001, Finn et al. [2] introduced the Body Text Extraction (BTE) algorithm to improve the accuracy of the content's classifier for digital libraries. They interpreted an HTML document as a sequence of word and tag tokens, and then extracted content by identifying a single, continuous region which contains the most words and the least HTML tags. Thus the algorithm tokenizes a page into either words or tags; the page is then sectioned into three contiguous regions, placing boundaries to partition the document such that most tags are placed into outside regions and word tokens into the middle region. This approach works well for single body documents, but it reversely destroys the structure of the HTML and does not produce good results for multiple body documents. In order for content of multiple body documents to be successfully extracted, the running time of the algorithm would become polynomial time with a degree equal to the number of separate bodies.

To overcome the limitation of BTE in discovering only a single continuous block of text, Pinto et al. [3] extended this method to construct Document Slope Curves (DSC), in which a windowing technique is used to locate document regions in which word tokens are more frequent than tag tokens. They used this technique to improve performance and efficiency for answering questions with web data in their QuASM system. Mantratzis et al. [4] presented an approach named Link Quota Filter (LQF) to identify link lists and navigation elements by identifying DOM elements which have a high ratio of text residing in hyperlink anchors. It can be applied to content extraction by removing the resulting link blocks from the document. The major drawback of this method is that it relies on structure elements, and it can only identify hyperlink-type noise.

McKeown et al. [5][6], in the NLP group at Columbia University, detects the largest body of text on a webpage (by counting the number of words) and classifies that as content. This approach works well with simple pages. However,

this approach produces noisy or inaccurate results handling multi-body documents, especially with random advertisement and image placement.

The Content Code Blurring (CCB) algorithm was introduced by Gottron[7] in 2008. Content regions are detected in homogeneously formatted source code character sequences. Weninger et al. [8] introduced the Content Extraction via Tag Ratios (CETR) algorithm, a method to extract content text from diverse web pages using the HTML document's tag ratios. The approach computes tag ratios on a line-by-line basis and then clusters the resulting histogram into content and noise areas. This is a laconic and efficient algorithm, however vulnerable to the page's source code style changes.

Buyukkokten et al. [9][10][12] define "accordion summarization" as a strategy where a page can be shrunk or expanded. They also discuss a method to transform a web page into a hierarchy of individual content units called Semantic Textual Units (STUs). First, STUs are built by analyzing syntactic features of an HTML document, like text contained within paragraph (<P>), table cell (<TD>), and frame component (<FRAME>) tags. These syntactic features are then arranged into a hierarchy based on the HTML formatting of each STU. STUs that contain HTML header tags (<H1>, <H2>, and <H3>) or bold text (<B>) are given a higher level in the hierarchy than plain text. The hierarchical structure is finally displayed on PDAs and cellular phones. While Buyukkokten's hierarchy is similar to our DOM tree-based model, DOM trees remain highly editable and can easily be reconstructed back into a complete web site. DOM trees are also a widely-adopted W3C standard, easing support and integration of our technology. The main problem with the STUs approach is that once the STU has been identified, Buyukkokten et al. perform summarization on the STUs to produce the content that is then displayed on PDAs and cell phones.

Kaasinen et al. [11], discusses methods to divide a web page into individual units similar to cards in a deck. Like STUs, a web page is divided into a series of hierarchical "cards" that are placed into a "deck". This deck of cards is presented to the user one card at a time for easy browsing. It also suggests a simple conversion of HTML content to WML (Wireless Markup Language),which results in the removal of simple information such as images and bitmaps from the web page so that scrolling is minimized for small displays. The problem with the deck-of-cards model is that it relies on splitting a site into tiny sections that can then be browsed as windows(cards). But this means that it is up to the user to determine on which cards the actual contents are located.

PROPOSED METHOD

A solution employs multiple extensible techniques that incorporate the merits and overcomes the demerits of the previous work on content extraction. To analyze a web page for content extraction, the web page is first passed through an HTML parser that corrects the HTML and creates a Document Object Model tree representation of the web page. The Document Object Model tree is hierarchically arranged and can be analyzed in sections or entirely, providing a wide range of flexibility for the extraction algorithm. The content extractor navigates the Document Object Model tree recursively, using a series of various filtering techniques to remove and modify specific nodes and extract only the content. Each of these filters can be easily turned on and off and customized to a certain degree.

There are two sets of filters, with different levels of granularity. The first set of filters simply ignores tags or specific attributes within tags. With these filters, links, scripts, styles, and many other elements can be quickly removed from the web page. This procedure of filtering is similar to conversion of HTML to WML. However, the second of type filters is more complex and algorithmic, providing a higher level of extraction than offered by the conversion of HTML to

WML. This set, which can be extended, consists of the advertisements remover, the link list remover, the empty table remover, and the removed links retainer.

The advertisement remover uses an efficient technique to remove advertisements. As the Document Object Model (DOM) tree is parsed, the values of the "src" and "href" attributes throughout the page are surveyed to determine the servers to which the links refer. If a link address matches against a list of common advertisement servers, the node of the Document Object Model (DOM) tree that contained the link is removed.

The link list remover employs a filtering technique that removes all "link lists", which are table cells for which the ratio of the number of links to the number of non-linked words is greater than a specific threshold (known as the link/text removal ratio). When the Document Object Model (DOM) parser encounters a table cell, the Link List Remover calculates the number of links and the number of non-linked words. The number of non-linked words is determined by taking the number of letters not contained in a link and dividing it by the average number of characters per word, which can be preset as say 5 (although it may be overridden by the user). If the ratio is greater than the user-determined link/text removal ratio, the content of the table cell (and, optionally, the cell itself) is removed. This algorithm succeeds in removing long link lists that tend to reside along the sides of web pages while leaving the text-intensive portions of the page intact.

The empty table remover removes tables that are empty of any "substantive" information. The user determines by making settings, which HTML tags should be considered to be substance and how many characters within a table are needed to be viewed as substantive. The table remover checks a table for substance after it has been parsed through the filter. If a table has no substance, it is removed from the tree. This algorithm effectively removes any tables leftover from previous filters that contain small amounts of unimportant information.

After examining many web article pages, we have found that there are basically three ways in which paragraphs are created in HTML examples: (1) text, (2) <p> text, and (3) text
 text. Note that both and <p> are block tags where each of these nodes forms a paragraph block on the rendered web page. The third type, text
 text, on the contrary uses the
 (line-break) tag to separate the text into paragraph blocks. In a typical article, there are usually more than one article subtrees that are candidates as the main article text-body. To determine which article subtree is the main article, we use the following rule-based algorithm. First, find all the article subtrees having large body of text by summing up all the text size (number of characters) in the paragraphs, and then use a threshold to only keep the subtrees with large text-sizes. A good threshold value is 500, which means that the minimum article should contain at least 500 characters. Each article subtree, visually, is a content block on the rendered web page. The article subtree or content block might contain other content type beside paragraphs such as images and hyperlinks to other web sites. If the first step produces multiple article subtrees, then we choose the one that is first occurring on the DOM, which visually should be the upper-most on the web page. Note that this article subtree might not have the largest text-size. The assumption is that the main article text block should always be on top of the page relative to other large text blocks.

Since we are using the image caption for the semantic similarity test, the image and its caption need to be identified and extracted. On the Document Object Model (DOM), it is easy to identify images, by simply using the HTML tag semantic, e.g. we look for the <img> tag elements. We assume that if there is an article image, it will be embedded in the article block, equivalent to having the image node in the article subtree. We further assume that if an image has no caption, it is a non-article image. Thus we only consider images with captions in the article subtree as candidates for similarity testing. By examining the DOM of many web pages, we found that the smallest bounding block that contains

both the image and its caption is also their first parent block-element. From this observation, the following heuristic rules are used to extract image caption: For each image in the article subtree, and the first parent block-element. All the text content in this parent element is considered to be caption text, which can now be used to conduct the similarity test with the article text-body.

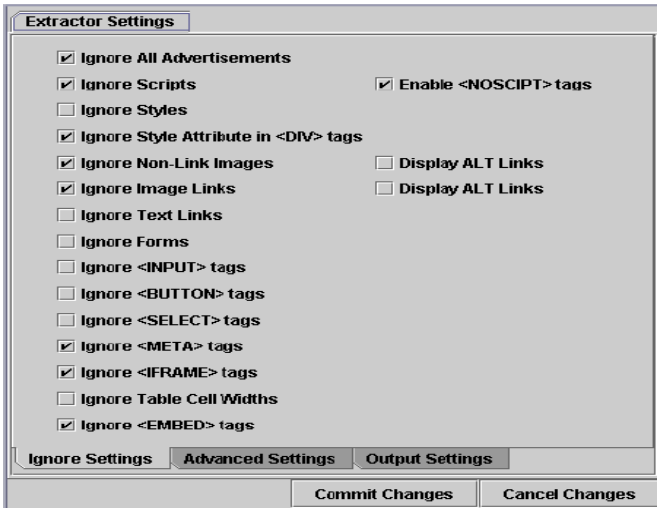

Figure 1: Filtering Settings Interface

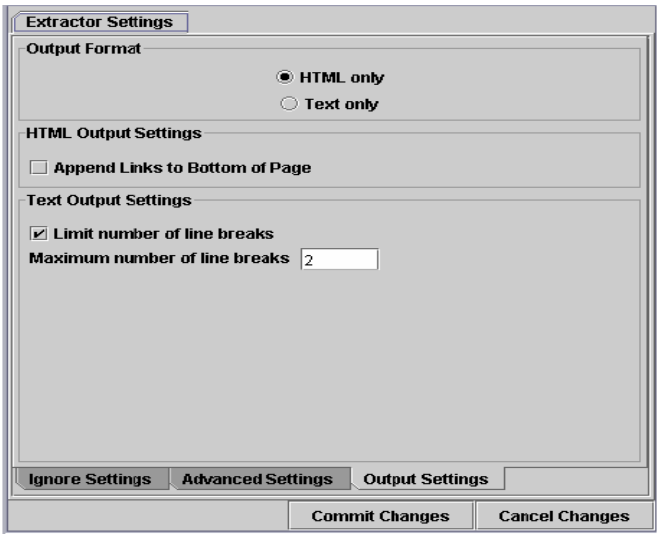

Figure 2: Output Settings Interface

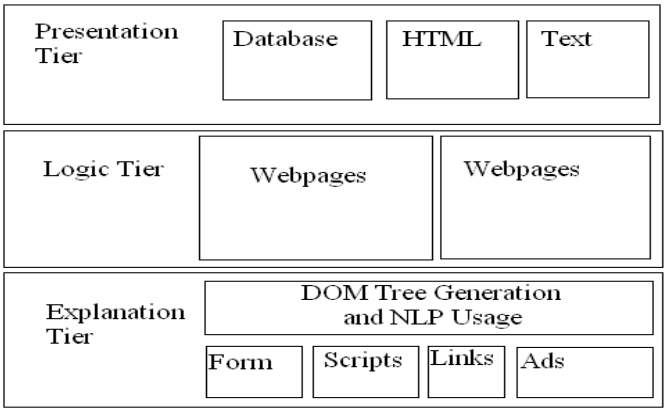

Figure 3: System Diagram

Text extracted from various text blocks as well as image captions are compared for semantic similarity in this step. The content similarity algorithm relies on NLP techniques for information extraction. The algorithm first performs Named Entity Recognition where names of people, places and organizations (commercial, governmental or NGOs) are extracted. To perform Named Entity Recognition leveraged GATE can be used. After the Named Entity Recognition is completed, the algorithm compares the extracted patterns of named entities in any two given content blocks to determine content similarity. For smaller content blocks that do not have high frequencies of named entities, it performs match of representative named entities that occur with high frequencies in the bigger content block. For large content blocks our algorithm calculates frequencies of each recognized entity in both content blocks. Then it normalizes frequency distributions and maps normalized frequencies into a multidimensional space with number of entities representing the number of dimensions. From that it derives a cosine similarity based on the two normalized frequency distributions from two text blocks. This represents the angle between two frequency distribution vectors. Orthogonal vectors would have cosine similarity measure 0 which represents no similarity. Perfect similarity would be indicated by vectors in the same direction, where cosine similarity measure would be 1.

Cosine Similarity Measure = $V_a.V_b$ /($|V_a|.|V_b|$) where, V_a and V_b are vectors representing normalized frequency distributions for the recognized named entities and V_a and V_b represent magnitudes of the vectors. It should be noted that since this approach relies on semantic similarity it will work even if any advertising images were placed in middle of the article.

While the above filters remove non-content from the site, the removed link retainer adds link information back at the end of the document to keep the page browse able. The removed link retainer keeps track of all the text links that are removed throughout the filtering process. After the Document Object Model (DOM) tree is completely browsed, the list of removed links is added to the bottom of the page. In this way, any important links that were previously removed remain accessible to the user.

After the entire DOM tree is parsed and modified appropriately, it can be output in either HTML or as plain text. The plain text output removes all the tags and retains only the text of the site, while eliminating most white spaces. The result is a text document that contains the main content of the site in a format suitable to be summarized, or for speech rendering or storage. Our algorithm not only finds the content, but also eliminates non-content. In this manner, we can still process and return results for sites that don't have an explicit "main body".

FUTURE SCOPE

The current implementation in Java uses a 3rd-party HTML parser to create DOM trees from web pages. Unfortunately, most of the publicly-available Java HTML parsers are either missing support for important features, such as XHTML or dynamically-generated pages (ASP, JSP). To resolve this, we intend to support commercial parsers, such as Microsoft's HTML parser (which is used in Internet Explorer), in the next revision. These are much more robust and support a wider variety of content. The integration will be accomplished by porting the existing proxy to C#/.NET, which would allow for easy integration with COM components (of which the MS HTML parser is one).

We can also work on improving the proxy's performance; in particular, we aim to improve both latency and scalability of the current version. The latency of the system will be reduced by adopting a commercial parser and by improving our algorithms to process DOM incrementally as the page is being loaded. Scalability will be addressed by

re-architecting the proxy's concurrency model to avoid the current thread-per-client model, adopting a stage-driven architecture instead.

As for future work, evaluation of this system on a large set of web pages would be targeted. In addition, optimizing content extraction and similarity measure using statistical techniques is amongst the main goals.

CONCLUSIONS

In this paper, a novel approach of content extraction has been described using NLP with HTML DOM to differentiate relevant images from advertisements images in web article pages and to extract relevant content. Working with the Document Object Model tree as opposed to raw HTML markup, helps to perform Content Extraction, identifying and preserving the original data instead of summarizing it. It is observed that most images in article web pages have textual cues such as caption around the images for which semantic comparison can be performed with the main text body to determine image relevance. The text body and caption extraction schemes are based on DOM analysis, and their accuracy is very high.

REFERENCES

1. F. R. Rahman, H. Alam, and R. Hartono. Content extraction from html documents. In *WDA2001*, pages 7–10, 2001.
2. Aidan Finn, Nicholas Kushmerick, and Barry Smyth. "Fact or fiction: Content classification for digital libraries". Smart Media Institute, Department of Computer Science, University College Dublin. August 11, 2002. <http://www.smi.ucd.ie/hyppia/publications/DELOS_workshopAF/DCUconf.html>.
3. D. Pinto, M. Branstein, R. Coleman, W. B. Croft, M. King, W. Li, and X. Wei. Quasm: A system for question answering using semi-structured data. In *Proceedings of JCDL '02*, pages 46–55, 2002.
4. Mantratzis, M. Orgun, and S. Cassidy. Separating xhtml content from navigation clutter using dom-structure block analysis. In *Proceedings of HYPERTEXT '05*, pages 145–147, 2005.
5. K.R. McKeown, R. Barzilay, D. Evans, V. Hatzivassiloglou, M.Y. Kan, B. Schiffman, S. Teufel. "Columbia Multi-document Summarization: Approach and Evaluation".
6. N. Wacholder, D. Evans, J. Klavans "Automatic Identification and Organization of Index Terms for Interactive Browsing".
7. T. Gottron. Content code blurring: A new approach to content extraction. In *Proceedings of DEXA '08*, pages 29–33, 2008.
8. T. Weninger, W. H. Hsu, and J. Han. Cetr – content extraction via tag ratios. In *Proceedings of WWW '10*, pages 971–980, New York, NY, USA, 2010.
9. O. Buyukkokten, H. Garcia-Molina, and A. Paepcke. "Accordion Summarization for End-Game Browsing on PDAs and Cellular Phones". In Proc. of the Conf. on Human Factors in Computing Systems. CHI'01, 2001.
10. O. Buyukkokten, H, Garcia-Molina, A. Paepcke. "Seeing the Whole in Parts: Text Summarization for Web Browsing on Handheld Devices". Proc. 10th Int. World-Wide Web Conf., 2001.

11. E. Kaasinen, M. Aaltonen, J. Kolari, S. Melakoski and T. Laakko. “Two Approaches to Bringing Internet Services to WAP devices”. In Proc. of 9th Int. World-Wide Web Conf. 2000. pp. 231-246.

12. O. Buyukkokten, H. Garcia-Molina, A, Paepcke, “Text Summarization for Web Browsing on Handheld Devices”, In Proc. of 10th Int. World-Wide Web Conf., 2001