Hybrid model of content extraction

Pir Abdul Rasool Qureshi, Nasrullah Memon *

The Maersk Mc-Kinney Moller Institute, University of Southern Denmark, Denmark

ARTICLE INFO

Article history:
Received 31 May 2011
Received in revised form 22 July 2011
Accepted 17 October 2011
Available online 3 November 2011

Keywords:
Content extraction
HTML
Open source intelligence
Information filtering

ABSTRACT

We present a hybrid model for content extraction from HTML documents. The model operates on Document Object Model (DOM) tree of the corresponding HTML document. It evaluates each tree node and associated statistical features like link density and text distribution across the node to predict significance of the node towards overall content provided by the document. Once significance of the nodes is determined, the formatting characteristics like fonts, styles and the position of the nodes are evaluated to identify the nodes with similar formatting as compared to the significant nodes. The proposed hybrid model is derived from two different models, i.e., one is based on statistical features and other on formatting characteristics and achieved the best accuracy. We describe the validity of model with the help of experiments conducted on the standard data sets. The results revealed that the proposed model outperformed other existing content extraction models. We present a browser based implementation of the proposed model as proof of concept and compare the implementation strategy with various state of art implementations. We also discuss various applications of the proposed model with special emphasis on open source intelligence.

© 2011 Elsevier Inc. All rights reserved.

1. Introduction

The quantity of information available today is more than at any point in history, but with this wealth of information comes even greater challenges. Unless we create the appropriate tools to deal with such information overload effectively, namely capable of automatically processing the online information, the benefits of having so much data at our fingertips simply disappear. The Web, as the largest database, often contains information that may be interesting for the researchers or the general public. The problem is that data is mixed together with formatting code, advertisement, or web page specific information such as navigation links; in that way that web page is human friendly, but the actual content is not machine friendly [6,8]. This mix of unwanted noise with the real content in a web page complicates the task of automatic information (content) extraction and processing. By the term “content extraction”, we mean the extraction of useful noise-free information from the web pages and is also known in the literature as “Information Filtering”. We have used the word “Content” and Information interchangeably in the rest of article because the real contents of the web page are actually the information which it provides in the context of Content Extraction.

Content Extraction is useful for visually impaired and blind. The idea is to identify the real content within a web page and increase the font size of the portions of the web page containing contents for better visualization or directly transforming the contents of the web page to speech. Natural Language Processing (NLP) and Information Retrieval (IR) can also benefit from content extraction as the models as they rely on relevance of contents and the reduction of “standard word error rate” for accurate results [11]. Presently, most of the NLP based IR applications require writing specialized extractors for each of
the web domain [11]. As generalized content extraction is less accurate than hand tailored extractors but is often found sufficient [11] and less laborious. Content Extraction is particularly useful for Open Source Intelligence (OSInt), which rely on automatic processing of open source information. The idea in this regard is to apply content extraction on web pages to extract information and then processing the extracted information to gain knowledge, all in automatic manner. Due to the increasing number of web pages and the amazing expansion of the Internet today, the area of content extraction is considered one of the most interesting in terms of data mining.

According to common observation, a typical web page contains a title banner, list of links in right or left or both for site navigation and advertisements, a footer containing copyright statements, disclaimers or even sometimes navigational links [27]. Mostly, meaningful content lies at the center of the page. As the web page design is not standard, which all web pages must comply, consequently, a more robust and flexible content extraction tool is essential.

The newer web pages are observed to have a cleaner architecture, providing separation among visual presentation, real content and the interaction layers [24]. The modern web pages have abandoned the use of old structural tags and adopted an architecture that makes use of the style sheets and \langle div \rangle or \langle span \rangle tags [27]. This is a welcome change because it eases the development and maintenance of web pages. However, it reduces the effectiveness of the old content extraction techniques, which rely on the presence of HTML cues like tables, fonts, sizes, lines, etc.

In this paper we propose a hybrid model named as “Content Vision” (CV). At first, it employs calculation of different statistical features associated with the different nodes of DOM tree to measure their usefulness in providing the information. DOM (http://www.w3.org/DOM) is the widely used standard for manipulating in memory representation of HTML content. The features like the quantity of text and the quantity of hypertext present at different nodes in the DOM tree are analyzed to determine the usefulness of the node in presenting the content. The calculation is based on the fact that the nodes associated with the content have higher values for the quantity of the text and lower values for quantity of hypertext. As these quantities may vary from a web page to other web page, so the quantities are normalized with respect to each page in order to achieve an optimal performance on different styled web pages. Once the statistical features are identified and nodes are classified as useful nodes, the nodes similar to useful nodes based on formatting characteristics and their position in the page are identified. All of the nodes classified as useful and nodes similar to useful nodes are considered to be the nodes containing real contents.

The experimental results verify the effectiveness of the CV model, as it achieved better results on varying web pages. Since, the proposed model does not make any assumption about the structure of the input web page, nor does it rely on the particular HTML cues, which increases its overall performance as compared to the other methods of content extraction.

The rest of the paper is organized as follows: Section 2 presents the state of art in the field of content extraction; whereas Section 3 presents the statistical model for extracting HTML contents. In Section 4, we discuss the implementation of the proposed model and compare our implementation strategy with the other implementations available in market. In Section 5, we describe the experimental setup and present the results, while Section 6 gives an account of various applications of the proposed model and its significance in open source intelligence. Section 7 concludes the paper and discusses the future directions.

2. State of art

The field of content extraction is well studied and a number of the methods have already been developed. The traditional approaches to extract content from HTML documents mostly emphasize on removing clutters, disabling javascript, removing images, etc. Examples include WPAR (http://sourceforge.net/projects/wpar), NoDoSE [1], XWRAP [16], etc. All of the discussed software include the human aided rule based techniques or regular expressions which are dependent on certain common web page designs and the elimination of advertisements by maintaining a blacklist of advertisers. An obvious disadvantage of the approach is that it is dependent on manual effort to create rules for each website and also update it as frequently as website updates. The term “Content Extraction” was first coined by Rahman et al. [2] along with the first and very basic algorithm. Finn et al. [9] discusses method (BTE) to identify useful information within the “Single Article” document, where the content is presumed to be in a single body. The method depends upon the tokenization of contents of the document into tags or words. The document is divided into three contiguous sections and boundaries are estimated in such a way to maximize the number of words within the sections. The method is useful for the documents having simpler layouts but does not produce the desired results with the dynamic content sites and complex layouts designed to make most of space available on a web page. McKeown et al. [18] find the body of a tag with the highest weight in terms of text present within the body. The detected body is considered to be the one containing useful information and the rest is ignored. Another method to extract contents is Document Slope Curves (DSC) [21], an extension to BTE algorithm to create Advanced DSC which employs windrowing technique to identify the regions in the document containing content. Mantratzis [17] used the amount of text within the anchor tags to identify the navigational lists. The idea was to remove the navigational links from the document and the approach is named as Link Quota Filter (LQF). Another similar algorithm is Largest Size Increase (LSI) [12] algorithm that works out the nodes of DOM tree which contributes most to the visible content in the rendered document. Deb Nath et al. developed Feature Extractor (FE) and K-Feature Extractor (KFE) [7] based on the block segmentation of HTML document. Another approach to extract content from HTML web page is Content Code Blurring (CCB) [10]. It works on identifying the maximum number of formatted source code characters to identify the content blocks. The most
The normalized deviation distinguishes main content areas, from the nodes like headers and advertisements, which are contributing less towards information being rendered by the page. However, this does not suffice to distinguish them. Fig. 2 illustrates the normalized deviation for the example web page. Nodes will yield very low values while the nodes used in rendering the most of the information have values nearer to 1. This effect is illustrated in Fig. 3, which shows the normalized deviation for the example web page.

We used Cobra (http://lobobrowser.org/cobra.jsp), an open-source implementation of a web browser to convert an HTML document. This approach employed supervised classification models and needed a collection of training documents in order to train the classification models. One of the examples of such strategy is Bar-Yossef et al. [3], which automatically detects the template of the web page by using the Largest Pagelet algorithm (LP). The weakness of the approach is that it depends upon the template of the web page and does not perform adequately on the data sets with the web pages of varying templates.

The proposed model is based on the structural analysis of HTML page and calculations of the quantity of text associated with different positions in the structure. For structural analysis the HTML document is converted into DOM tree, which is the standard practice for accessing and manipulating HTML documents and almost all of the HTML parsers implement it. We used Cobra (http://lobobrowser.org/cobra.jsp) an open-source implementation of a web browser to convert an HTML document into respective DOM tree but the proposed model is completely independent of the service and can use any implementation of DOM. The proposed model recognizes the layout of the page by comparing the quantity of text across each unique node in the DOM tree with the arithmetic mean of the quantity of text across all of the nodes in DOM tree. The deviation in text quantity at each node from the arithmetic mean represents how the node contributes towards the information being rendered to the user. The higher the deviation, the more information is rendered through that node. Fig. 1 shows the normalized deviation obtained from Eq. (2) identifies the less informative nodes of DOM tree. The less informative nodes will yield very low values while the nodes which are used in rendering the most of the information have values nearer to 1. This effect is illustrated in Fig. 3, which shows the normalized deviation for the example web page.
Fig. 1. HTML rendering of an example web page from BBC news site through Content Vision Browser.

Fig. 2. Deviation in Text Sizes for the BBC web page.

Fig. 3. Normalized Deviation in quantity of Text Sizes for the BBC web page.
from the informative nodes of DOM tree that have less quantity of information as for example the titles of the columns in tables, headings. To make this distinction clear, we calculate the link densities associated with the nodes of DOM tree along with their deviations. The term link density for any DOM tree node can be defined as the ratio of “the number of words that are serving as a hyperlink to some other document or other part of the same document” to “the number of words” within the boundaries of a single node of the DOM tree. The link density $L(i)$ for any DOM node $i$ can be calculated as

$$L(i) = \frac{l(i)}{a(i)},$$

$$NL(i) = \frac{L(i)}{\text{Max}(L(T))},$$

where $l(i)$ is the quantity of anchor text. Again the link density at a particular node can be divided by maximum link density within the page to normalize the results. Both link densities and normalized link densities for the BBC web page are shown in Fig. 4 and Fig. 5 respectively. By comparing the $NL(i)$ with $N(i)$ at any DOM node, we detect that whether the DOM node contributes towards information or it is there for navigation. For the advertisements, headers or footers in any page, the link densities are usually high values. This behavior is mainly because of the fact that the quantity of text within the DOM nodes which are part of headers, footers or advertisements is less and the quantity of links is more than the informative nodes of the DOM tree of the same page. Fig. 6 plots the deviations and link densities of the BBC web page against the DOM nodes. The proposed model incorporates these statistical features extracted from any DOM node to classify its informative or non-informative nature. The nodes which have higher link densities are just ignored as they contain more of the words as links than the number of words which are providing the information. Similarly the nodes having positive values for normalized deviation are the nodes which are more than average informative nodes and hence considered as useful. Generally, the difference of $N(i)$ and $NL(i)$ represents the usefulness of the node in providing the real content within the web page. The nodes with negative values are ignored, because they have more hyperlinks $NL(i)$ than information $N(i)$. Fig. 6 shows the comparison of $N(i)$ and $NL(i)$ for the BBC web page. For example nodes 24 and 71 have very high values for $NL(i)$ and comparatively low values for $N(i)$, which implies that the difference of $N(i)$ and $NL(i)$ will be less than 0. Our CV model rejects such nodes in order to filter out informative nodes from the web page.

Once the usefulness of nodes is calculated, the styles corresponding to the useful nodes are identified. Such styles are used as a guide to understand the template of the document and serve as a backup mechanism to extract content from the nodes which have sometimes uncommon statistical values. For example, the nodes with small quantity of text and high link density, but they are rendered with the same formatting styles as the nodes which have higher quantities of text associated, can be considered useful in content extraction, even when the difference of their $N(i)$ and $NL(i)$ is less than 0.
Our experimental results show that considering formatting characteristics is very useful, but very resource consuming and expensive operation. It is due to the fact that run time script execution and cascading style sheet expressions need to be resolved in order to calculate how the node will be rendered. This complicates the process of content extraction and consumes few more CPU cycles.

Fig. 7 illustrates the working of the CV content extraction model. It describes that the overall process involved in extracting the contents of the web page, the main steps involved in the process are described below:

- The User initiates the process by requesting web resource from browser GUI.
- BrowserGUI fetches the response, and Cobra Script Execution Engine is used to execute the runtime scripts.
- DOM tree extracted from the documents along with the associated styles and other dynamic values which are generated as a result of script execution.
- The Statistical Feature generator evaluates the DOM tree and generates the values of different features like Link Density and Quantity of Text.
- Estimation of usefulness of the node.
- Extracting styles and formatting characteristics associated with the useful nodes.
- Collecting the useful nodes and the nodes with similar styles as of useful nodes.
- Extracting contents from original DOM tree against nodes short listed in previous step.
- Generating output in plain Text and HTML format, this is rendered through browser before user.
These steps and their sequence of execution are depicted in Fig. 8, to illustrate the sequence of execution of these steps. The proposed model is an extended version of basic Content Vision Model [23]. The basic version emphasizes only on statistical features where as extended version also includes formatting and style oriented features. The next section describes the implementation strategy of the discussed model and explains the use of different open source APIs’ like Cobra.

4. Content Vision (CV) implementation strategy

The most common implementation strategy is to implement content extraction model as a web proxy. This allows centralized control to set up the extractor and provide content extraction services to a group rather than an individual. The proxy is coupled with a graphical user interface (GUI) to customize its behavior. This also eases the user from having the headache of installing and managing the content extractor. The example of such implementation is CRUNCH [11].

We devised a different strategy for implementation. We opt to extend an open source implementations of browser i.e. “Swing Web Kit” (http://www.genuitec.com/about/labs.html) and “Cobra” (http://lobobrowser.org/cobra.jsp). The former has more robust presentation engine and graphical user interface, while the other has robust script execution and HTML parsing capabilities. We named our extension as Content Vision (CV) and can be downloaded from (http://130.225.157.115:8080/ContentVision). It has all basic functionalities, which are necessary to serve as a proof of concept for proposed content extraction model. The contents extracted from sample BBC page (Fig. 1) by our implementation of CV model are shown in Fig. 9. We chose to implement the content extraction as a part of browser because of the following benefits:

- As a part of browser, it enables the execution of scripts, and Ajax requests, thus, is aware of the dynamic contents. All of the approaches discussed so far miss run time generated content which is created as a result of dynamic script execution.
- It enables the user controlled behavior to the implementation, that is, switch over to normal and content extraction mode is easy and controlled by user instead of administrator managing proxy.
- It mitigates the risk of missing the important disclaimers and copyright statements, which are considered non-content nodes, hence, wiped off from the content by web proxy based implementations. In the case, it is implemented as browser, the user has to go to the original page before content extraction, which makes it possible for user to have a view of such legal notices, and then if user choose to extract contents, it is according to his/her own wish. Same can be communicated to the user in the license agreement before he/she starts using the browser based content extractor.
5. Experimental evaluation

We conducted a set of experiments on different data sets to evaluate the performance of the proposed model implemented in CV browser. We used two data sets. The first data set includes test and training data provided with CLEANEVAL task (http://cleaneval.sigwac.org.uk/). CLEANEVAL is a shared task and competitive evaluation on the topic of cleaning arbitrary web pages, with the goal of preparing web data for use as a corpus, for linguistic and language technology research and development. The other data set MSS [20] comprises of news web pages. The data set has two collections of web pages. First collection is called Big5, it contains data from 5 news websites i.e. New York Post, Tribune, Techweb, Suntimes, and Freep. The other collection is Myriad40 containing randomly chosen pages from Yahoo! Directory. We used the same traditional precision, recall and F1 measurement metrics as used in most of the related study (for example [27]) for evaluation of the proposed model. Table 1 lists the F1 accuracies of CV model and other related models. It is worthy to mention here that in our experiment we included only CV model, the rest of values are taken from [27] and are presented for the sake of completeness and to reflect the overall picture of the state of art. The results revealed that the proposed CV model achieved the highest overall accuracy of 94.74
6. Applications

The presented model is best applied in the open source intelligence domain [26] and for knowledge gathering systems for counterterrorism [28]. The world today abounds in open information to an extent unimaginable to intelligence officers of the Cold War [19]. The present age's increasingly voluminous open source intelligence sheds light on issues of the day for all source analysts, covert collectors, and policymakers, but it has not been fully incorporated to be that effective [26]. The reason is: "Collecting intelligence these days is at times less a matter of stealing through dark alleys in a foreign land to meet some secret agent than one of surfing the Internet under the fluorescent lights of an office cubicle to find some open source. The world is changing with the advance of commerce and technology. Mouse clicks and online dictionaries today often prove more useful than stylish cloaks and shiny daggers in gathering intelligence required to help analysts and officials understand the world. Combined with stolen secrets, diplomatic reports, and technical collection, open sources constitute what one former deputy director of intelligence termed the “intricate mosaic” of intelligence” [25]. This extract sheds light on the importance of open source information. Open source information is mainly available in HTML format, where proposed content extraction model can prove very useful. We take an example of Early Warning System (EWaS) [22]; aimed to predict terrorist threats and generate early warnings in case any suspicious activity is observed while monitoring open source information. EWaS model relies on data collection from open data sources, information retrieval, information extraction for preparing structured workable datasets from available unstructured data, and finally carrying out detailed investigation. EWaS system used the presented content extraction model for acquiring and then filtering the noise from HTML data sources. Qureshi et al. [22] discussed the results of an experiment conducted to evaluate and analyze the information extracted from open source information against a thwarted terrorist plot (known as Bonjika). In the experiment, EWaS system estimated the command structure (hierarchy) responsible for the terrorist plot [22]. The results (estimated command structure) of the experiment were found to be consistent with real facts which indicate that none of the terrorist nodes were missed in the content extraction process. The content extraction process was carried out on approximately 3000 web pages with random layouts structured, unstructured, single body, and multiple bodies in a single layout. This experiment shows that the proposed content extraction is valid over a variety of input data sources. The model also fits in the process where the data is needed to be transported over an expensive medium such as mobile phones or PDAs. In such cases, the proposed extraction model can be applied to filter markup text to save bandwidth. In the discussed experiment with the terrorist information, it was observed that nearly 20 to 40 of the DOM tree nodes contained useful information and the rest of the tree nodes were only for layout and formatting styles; this finding supports the validity of the model for saving bandwidth. The proposed model can also be used in filtering useful content from HTML pages and transforming it into speech for visually impaired or blind people [11]. The content extraction can eliminate the markup tags which are not needed in the speech transformation.

7. Concluding remarks and future work

We presented a statistical content extraction model, implemented as an extension to the browser. The presented model achieved the best 94.74 accuracy on different data sets comprising of a variety of web pages with different layouts. We also specified the benefits of our implementation of content extraction over the traditional web proxy based implementation. Finally, we described different applications of content extraction and its significance in open source intelligence. In nutshell, the following contributions can be claimed in our paper:

- Introducing a novel hybrid model for content extraction which is based on the combination of statistical features associated with a node in DOM tree and formatting characteristics. To the best of our knowledge, the best effective models like CETR or VIPS both works on each of statistical features like Tag ratio or formatting characteristics respectively.
- We coined the idea of implementing the content extraction from within browser and discuss its benefits over existing implementations, mainly as web proxy. The proposed implementation is made available for download and use.
- We discussed the legal and privacy related implications associated with the content extraction. We believe that this perspective of viewing content extraction is not yet discussed.
- The experiment conducted on standard data sets supports our argument that combining formatting characteristics and statistical features in content extraction model achieves promising results.
- The experimental results also support the fact that dynamic script execution and stylesheet elements are significant in the process of content extraction. The existing implementations of content extraction ignore scripts and other dynamic content or style generation technologies.

We discuss how the content extraction can be effectively used in real time projects like EWaS and in gathering open source intelligence. We aim to apply the presented content extraction model with the goal to prepare web data practically for use as a corpus, for linguistic and language technology research and development. We plan to modify the existing implementations of supervised and unsupervised models to directly consume HTML data as training and testing data. The other two practical applications, where we aim to utilize our content extraction model are “HOBO: browser for visually impaired” and “AMCG: a dynamic mobile code generator as an extension to web server”. The HOBO browser is aimed to read out (Speech generation) contents for visually impaired and AMCG will automatically optimize the web pages for rendering on smart phones and
PDA’s. Presently, websites mainly develop different versions to get smartly rendered on mobile phones. AMCG is aimed to automate dynamic code generation for such applications.

References


