CUTER: an Efficient Useful Text Extraction Mechanism

George Adam, Christos Bouras, Vassilis Poulopoulos
Research Academic Computer Technology Institute, Greece and
Computer Engineer and Informatics Department, University of Patras, Greece
adam@ceid.upatras, bouras@cti.gr, poulop@cti.gr

Abstract— **In this paper we present CUTER, a system that processes HTML pages in order to extract the useful text from them. The mechanism is focalized on HTML pages that include news articles from major portals and blogs. As useful text we define the body of the article that contains the news report. In order to extract the body of the article we deconstruct the HTML page to its DOM model and we apply a set of algorithms in order to clean and correct the HTML code, locate and characterize each node of the DOM model and finally store the text from the nodes that are characterized as useful text nodes. CUTER is a subsystem of peRSSonal, a web tool that is used to obtain news articles from all over the world, process them and present them back to the end users in a personalized manner. The role of CUTER is to feed peRSSonal with the body of the. In this paper we present the basic algorithms and experimental results on the efficiency of the CUTER text extractor.**

Text extraction, HTML analysis, DOM analysis, useful text

I. INTRODUCTION

The World Wide Web is considered nowadays to be one of the basic sources for information gathering and data searching. It is inevitable that more and more people go online in order to fulfill many of their everyday tasks, one of which is news reading. While many major and minor news portals and blogs offer a huge amount of information for reading purposes, their web pages are often cluttered with distracting features around the body of an article that prevent the users from the actual content they are interested in. These features may include pop-up ads, flashy banner advertisements, unnecessary images, or links scattered around the screen. A solution to the aforementioned problem is a system that is able to delete the information that is of no user interest. The idea is to automatically extract the text or generally the content that seems to interest the user or delete any content that may distract the internet users. Automatic extraction of useful and relevant content from web pages has many applications, ranging from enabling end users to accessing the web more easily over constrained devices like PDAs and cellular phones to providing better access to the web for the visually impaired.

There is a large body of related work in content identification and information retrieval that attempts to solve similar problems using various techniques. Finn et al. [1] presents some methods for extracting content where the sources are single-articles and their content is a simple single body. The efficiency of this approach is proved for single-body documents; still, the HTML [11] structure is totally destroyed. Additionally, the approach is inefficient for multi-body documents, where content is segmented into multiple smaller pieces, something very common on Web logs ("blogs").

McKeown et al. [2] is presenting a different method where the largest body of text on a webpage is detected and classified as content. Similarly to the previous method this approach seems to be efficient for simple pages and pages with single-body articles. The algorithm produces inaccurate results when handling with multi-body documents, especially with random advertisement and image placement. A similar procedure with analogous outcomes is proposed by Wacholder et al. [3].

An interesting technique is proposed by Rahman et al. [4] that use structural analysis, contextual analysis, and summarization in order to extract the useful text out of the HTML page. This approach has more limitations than the flexibility of implementation as it does not include efficient algorithms for summarization and it is inefficient when merging sections of multi-body documents.

Gupta et al [5] work with Document Object Model [10] tree and perform content extraction in order to identify and preserve the original data. This method is effective for content extraction and manages extract useful data when parsing single-body pages. However, it is not focused on articles' extraction, thus the output is likely to contain useless content when parsing multi-body pages.

Similar to the useful text extraction mechanisms can be considered systems that are designed for web clipping. Web Clipping services target users who have specific information needs, but lack the time or knowledge to search and browse the Web. Three of the most representative efforts on web clipping services are WebCQ [6], THOR [7] and IEPAD [8]. WebCQ is a server-based change detection and notification prototype system for monitoring changes in arbitrary web pages. THOR is a scalable and efficient mining system for discovering and extracting QAPagelets from the Deep Web. QAPagelet is used to refer to the content region in a dynamic page that contains the result to the query performed. IEPAD is an information extraction system designed to recognize patterns in Web documents.

Our system, CUTER, is a useful text extractor mechanism that is focalized on extracting the body of articles and generally information that exists on a web page and is written in article-style (title and body). The difference of our system compared to the already existing systems is that it achieves to extract a large amount of the original text from both single-body and multi-body documents and additionally it is able to extract multimedia that accompanies the article. Moreover, our mechanism as part of a multi-purpose system is able to utilize "knowledge" that derives from parts of the peRSSonal system that are executed before CUTER starts the useful text extraction.

The rest of the paper is structured as follows. In the following section we present our motivation and in the next section we present the architecture of CUTER mechanism. In the fourth section we analyze the algorithms utilized within the mechanism. Following we analyze some experimental evaluation that is done in order to present the accuracy of the mechanism and we conclude with remarks and future work on the system.

II. WHY PERSSONAL

The concept behind CUTER as part of peRSSonal [9] is the design of a single web place that offers, in a unified way, personalized and dynamically created views of news deriving from RSS feeds. The internal procedure of the system before an article reaches the end user is: first, all news and articles should be collected in real time from major news portals and blogs. At this stage CUTER is generated as we need to extract the text of the articles from the collected HTML pages. Furthermore, analysis of the extracted text with text pre-processing techniques is applied while categorization and summarization follows. Finally, we present the information to the end user assuring a personalized view, free of useless extra information, fitting in this way the user's demands; fulfilling both their personal profile and their device capabilities. Figure 1 depicts the above procedure. This manuscript focalizes on the module of the system that applies the useful text extraction on the collected HTML pages.

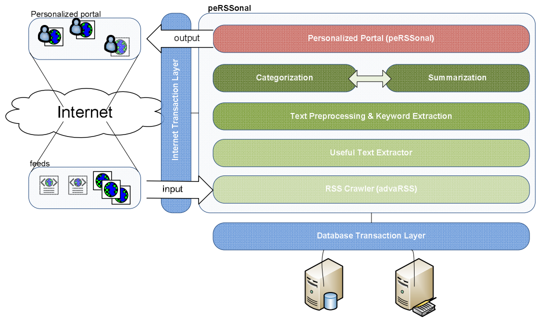
Personalized portal
peRSSonal
output
Personalized Portal (peRSSonal)
Categorization
Summarization
Internet
Internet Transaction Layer
Text Preprocessing & Keyword Extraction
Useful Text Extractor
feeds
input
RSS Crawler (advaRSS)
Database Transaction Layer

Figure 1: peRSSonal's architecture.

III. ARCHITECTURE

CUTER is a standalone system by means of input and output. It receives a universal XML format as its input and it produces XML output. This makes the mechanism flexible and easily adopted by mechanisms that can feed CUTER with HTML code and can be fed by it with text and multimedia.

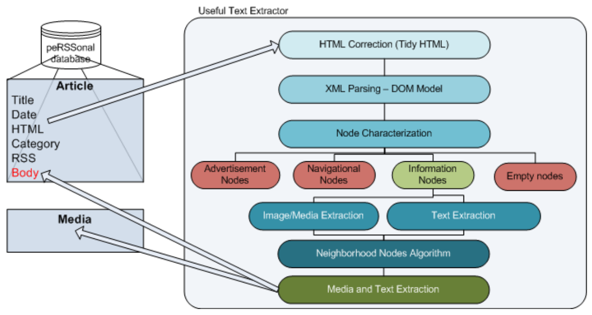

Figure 2: CUTER Architecture

The previous figure presents the architecture of the system. As already mentioned in our introductory part, CUTER is part of a multi-purpose system and thus, some modules of the systems are preceding and some others follow. The system that precedes is an RSS/HTML crawler that extracts the HTML code of the articles located into RSS feeds. The input of CUTER is pure HTML code.

A first module receives the HTML code from the articles deriving from the RSSs and tries to tidy it. This procedure is essential in order to correct all the HTML tags as all the browsers allow the HTML programmers to make mistakes without any pay back. The correction will help us create an efficient DOM model of the HTML page. The DOM model is an abstract tree structure of the HTML code with each node of the tree representing the HTML tags. In order to correct the HTML code we utilize HTML Tidy Open Source Project [12].

After the creation of the DOM model, the node characterization module starts to process the nodes of the tree in order to characterize them. Four possible characterizations exist in CUTER: (a) advertisement, (b) navigation, (c) information, which is further separated into (c_1) text and (c_2) multimedia nodes, and (d) empty. The text information nodes (c_1) are the ones that could possibly form the body of the article while multimedia information nodes (c_2) will include useful multimedia for presentation together with the article.

The characterization of the nodes will lead to sets of text nodes which will be processed by the neighborhood nodes algorithm in order to finalize on the exact text that will be extracted by the system as the article's useful text. At this stage of the system any nodes that are nested into c_1 nodes and include multimedia are characterized as c_2 nodes.

IV. ALGORITHM ANALYSIS

Useful text extraction is a procedure that includes isolation of parts of an HTML page (HTML is referring to the DOM model). In our occasion the HTML page is a page that includes a single article that can be single-body article or multi-body article. The following figure depicts a single-body and a multi-body article in order to understand the difference.

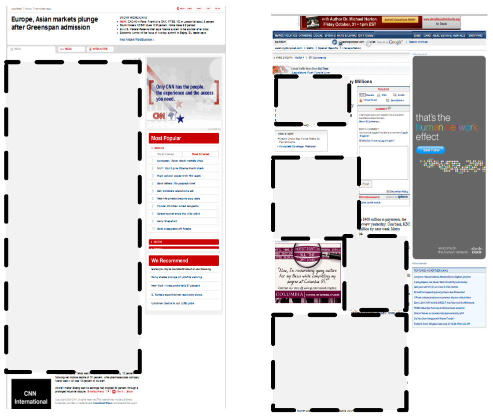
Figure 3: **Single-body and multi-body articles**

The analysis and extraction of the, what is called, useful text relies on the way that a plethora of pages is constructed and on the DOM model in which an HTML page can be decomposed.

As already mentioned, the implementation of the mechanism is based on the decomposition of the HTML structure into the DOM model. This means that we create a tree of the HTML page whose nodes are tags, and the leafs are pure text, images or any other component that can be presented in an HTML page (video, sound, flash and more). By obtaining the text of every leaf of the tree we assume that we have all the text of the HTML page. A first approach could be to get all the text that derives from every leaf and assume that it forms the text that we need. But it is obvious that in this case we do not care about the text that is included on parts of the page that form a menu, or an advertisement or even a tag of some image or other multimedia. In order to achieve better extraction of the useful text, or in other words the main body of the articles, we have to define specific attributes for this text.

By examining the web pages and the decomposed HTML page we conclude to the following attributes about the parent node of the leafs with useful text.

First of all, nodes with useful text have a high ratio of text (size in bytes) compared to the whole text of the webpage. Second, the neighbour nodes of the aforementioned nodes usually have the same high ratio. These neighbours can also be part of the useful text. Third, specific HTML tags are found with higher ratio in these leafs. More specifically much more text formatting tags (<b>, <strong>, <i>, <em>, <p>, etc.) can be found in leafs with useful text rather than other nodes. Fourth, the useful text is more likely to be after the article's title (note that we already know the article's title deriving from the RSS analysis of the module that precedes CUTER). By combining the four assumptions we define variables for each of them and apply a repetitive algorithm on the nodes of the DOM model in order to characterize each node. The variables that we use and will be used later in this text are

Table 1: Variables utilized in CUTER's algorithmic procedure

Name	Description
SL_x	The size in bytes of the text for the leaf X
SH	Represents the size bytes of the full text. It is the sum of all the SL_X.
SA_x	The size in bytes of the text for the leaf whose parent is an A tag (link).
IX	The identifier of a node

In order to recognize a leaf as part of the useful text of the HTML page we define specific conditions under which a leaf is named as a text node. In order to achieve this we define LP and TP, where:

$$LP = SA_x / SL_x \quad (1)$$

$$TP = SL_x / SH \quad (2)$$

LP is the link percentage of a node while TP is the text percentage of a node. High link percentage is translated into a node that contains lots of links and thus is difficult to be a useful text node. Nodes with high percentage of links usually form navigation menus, advertisements and generally features of the HTML page that are more likely placed for browsing purposes rather than for information purposes.

High values of the TP can be translated into nodes that can possibly form the useful text. The high values of the TP cannot be directly translated into nodes that hold useful text. Further processing is needed especially for multi-body articles.

At first we omit all the nodes (and their children) with high link percentage. Experimental evaluation on several web pages proved that the limit of the LP should be placed at 50%. This means that when half of the text of a node is text deriving from <A> tags then the whole text of the node is useless. On the contrary, the TP does not have a constant limit. We measure the TP of all the nodes of the tree and sort them according to their ID. Starting from the node (nk) that has the higher TP we measure the relative distance between nk and all the other nodes. Starting from nodes (nx) that are close to nk, if their TP is high (greater than TP(nk)/3) we unite nk with nx and repeat the procedure. The rest of the nodes are omitted.

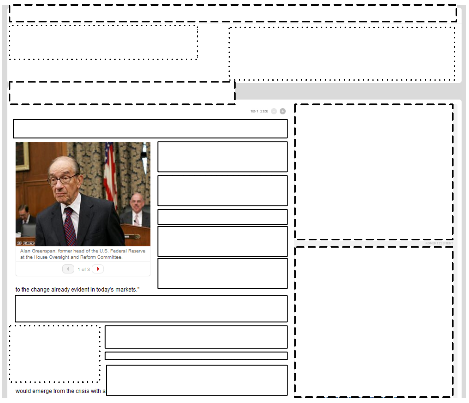
Figure 4: **Parts that are omitted from the beginning (dashed lines), candidates of useful text but finally omitted (dotted lines), extracted useful text (normal lines)**

This procedure will lead us to keep nodes with high TP that are neighbours. The previous figure represents the selections that are made for the characterization of the nodes of an HTML page. The dotted-line squares depict nodes that were "candidates" but finally omitted while the dashed-line squares depict nodes that were omitted due to high LP.

Finally, what is selected is depicted in the normal-line squares.

The algorithm that is utilized in order to extract the normal-line areas of the web page that usually form the useful text is the following.

```
    //We keep only nodes with some content
    DeleteAllEmptyNodes(xml);
    //SH holds the text length of the whole HTML page
    SH=length(getAllContent(xml));
    nodes[]=getAllNodesDepthFirst(xml);
    nextNodeID=1;
    for_each(nodes[] as currentNode)  {
    //Get the text length from current node and children
nodes
    SLx=length(getAllContent(currentNode);

    //Get the text length from all children A tag nodes
    SAx=length(getContentFromAtags(currentNode);

    LP=SAx/SLx;
    TP=SLx/SH;
    IX=nextNodeID++;

    //A valid node contains high text percentage and low
link percentage
    if(LP<maxLP AND TP>minTP)  then
     addToValidNodes(currentNode);
    }
    //Merge neighbouring nodes to one node using the node
ID number
    groupedNodes[]=mergeNeighbouringNodes(validNodes[]);
    maxLength=findTheMaximumTextLength(groupedNodes[]);
    acceptableLength=AL * maxLength;
    //The article is usually the first node with big
enough content
    article=firstNodeWithLengthGreaterThan(acceptableLengt
h);
```

In the code above we define maxLP as the maximum acceptable link percentage (as already mentioned is 50%), minTP as the minimum acceptable text percentage (depending on each webpage) and AL as the factor that is multiplied with the maximum found text size of a node in order to avoid content that is derived from "comments". in multi-body pages.

The algorithm first deletes all the nodes that do not have text. Afterwards, we define SH as the total size in bytes of the whole text of the page. A repetitive procedure begins and we measure SLx and SAx in order to define LP and TP. The definition of LP and TP leads to characterization of nodes. Following this procedure we group the nodes according to the neighboring algorithm and produce the final extracted text.

V. Experimental Evaluation

As the system seems to have much difference from systems that exist we are not able to make a comparable experimental evaluation with other mechanisms. In order to be more precise, our system monitors RSS feeds and when a new article occurs it visits the article's page and downloads the code. At this stage, CUTER is responsible for parsing the HTML code and extracts all the useful information which can be text and multimedia.

In order to conduct our experiment we observed the possible conditions of the articles structure. We define three different conditions under which an article is presented: (a) single body article, (b) multi-body article and finally (c) either of the aforementioned together with comments attached at the end of the article. A single body article is an article which text is not interrupted by advertisements or useless comments. A multi-body article is everything that is not single body and more specifically it is articles whose text is interrupted by advertisements or any other useless (for the internet user) information. Finally, as more and more websites enable commenting on the articles by the internet users a new category is also present and it is articles (single or multi-body) followed by the user's comments.

At the following experiment, we determine the accuracy of the mechanism using two values. The first value is the percentage of the article that was extracted. This value equals to 100% when the output contains all the useful text and 0% when it is not contained at all. The second value is the percentage of the unwanted content that is present in output. As unwanted content, we define any part of the page that doesn't belong to the article, such as advertisements, unrelated images, links, etc. The extraction process is accurate when the output is exactly the same with the useful content, which means that the first defined value must be 100% and the second 0%.

The mechanism was tested with the three different aforementioned forms of pages. The database includes more than 100.000 articles and the experiments were conducted for more than 1500 articles of each type. As it is expected, the system is able to operate with very high accuracy for single body pages as it is obvious from the next figure.

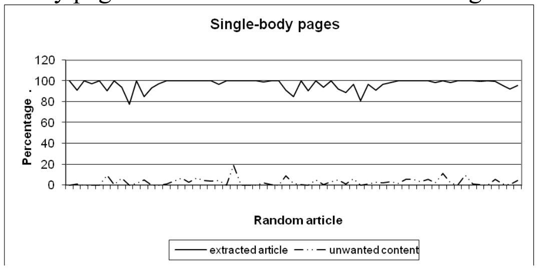

Figure 5: Single body pages

For the second type of pages, the article is segmented inside the page. Between two segments there is unwanted content that must not be extracted.

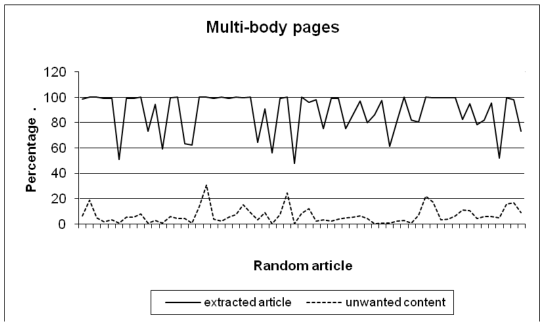

Figure 6: Multi body pages

From the following figure it is obvious that some useful content is "lost" and the output contains only some segments of the article and not the whole article.

It is important to note that in every occasion the article is retrieved from its beginning and usually the parts that are lost are after the first third of the size of the article.

The last form includes pages that contain user comments, in most cases below the article. A comment is usually related to article's topic, thus it is difficult to recognize that it is not a part of the useful text.

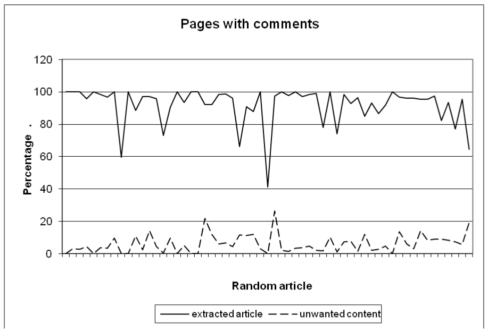

Figure 7: pages with user's comments

From the experiments it is obvious that the system is able to produce efficient results for single-body pages and it seems to add noise for multi-body pages and pages with user comments. Usually, the multi-body pages have the main body segmented with multimedia or advertisements separating the parts of the main text. In these cases, the system either adds noise to the extracted text by including not useful text or it decides that it found the end of the text, thus stopping the whole procedure.

The following table presents the accuracy and noise of the system in the three different types of pages. The accuracy (useful data) is the fraction of the text size of the article that is extracted to the actual text size of the article. As noise (useless data) we define the fraction of the text size of the useless information that is extracted compared to the text size that is extracted. If X is the text size extracted, Y is the useful text and A is the original articles then accuracy is defined as Y/A and noise is defined as (X-Y)/X = 1 – (Y/X).

Table 2: Accuracy and Noise of the system

	Useful data	Useless data
Single body articles	96,71%	3,40%
Multi-body articles	90,46%	6,55%
Articles with user comments	91,91%	6,06%

As it is obvious from the previous table the system can perform with very high accuracy and low noise in almost every situation. In the first type of articles the system is able to extract more than 95% of the original article while the extracted data consists of 3% of useless data. In the other two occasions the system is able to extract more than 90% of the original article but the extracted data consists of more 6% of useless data. By means of text size the usual size of an article is about 1500 characters which is translated to 250 words for the English language. At the worst occasion (multi-body articles) the system is able to retrieve correctly more than 1300 bytes of the text which is more than 220 out of the 250 words. For larger articles, of around 5000 bytes, which is 800 - 1000 words, the system is able to retrieve 720 to 900 words respectively. At the same worst occasion 40 to 60 words out of the total words extracted seem to be noise. As long as CUTER is part of a multi-purpose system that, among others, applies summarization on the extracted text we discovered that the summarization algorithm is able to omit any noise as it can define it as text irrelevant to the real subject. Finalizing, for the purpose that CUTER is constructed for the efficiency is assumed to be 100% on the end result something that is obvious on the outcomes to the peRSSonal web portal that presents the articles back to the users, where no noise is discovered in the summarized text.

VI. Conclusion and Future Work

We presented CUTER a useful text extraction mechanism that is fed with HTML pages including news articles and extracts the main text out of it. CUTER is part of a larger system, peRSSonal, that implements article's fetching, analysis, keyword extraction, text categorization and summarization and finally, presents the outcomes of the aforementioned procedures in a personalized way through a web portal. CUTER is a crucial part of the system as it is the main feeder of text and multimedia to the next procedures and thus its behavior can affect the whole system. The results of the accuracy and noise of the system shows that more than 90% of the text acquired is part of the real article body of the HTML page. Moreover for pages that have single body the mechanism achieves to acquire more than 95% of the real body. Additionally, as peRSSonal applies more algorithmic procedures on the extracted text we finally achieve 0% noise on the summarized text as the summarization sub-system is able to recognize the irrelevance between the actual body of the article and any noise added.

For the future we would like to further enhance the module of CUTER that extracts multimedia from the articles in order to have higher accuracy as this part of the system is still in experimental procedure. Moreover, the system can be further enhanced with lexica in order to correct the text that is extracted before it is presented to the next sub-systems of peRSSonal mechanism. Additionally, parts of the summarization procedure (and generally the text processing procedures) can be applied on CUTER in order to be able to understand the difference in meaning between the actual body of the article and any irrelevant (noise) information that is fetched. This change can further enhance the accuracy on results for multi-body articles and articles with user comments.

References

[1] Aidan Finn, Nicholas Kushmerick and Barry Smyth. "Fact or fiction: Content classification for digital libraries". In Joint DELOS-NSF Workshop on Personalisation and Recommender Systems in Digital Libraries (Dublin), 2001.

[2] K.R. McKeown, R. Barzilay, D. Evans, V. Hatzivassiloglou, M.Y. Kan, B. Schiffman and S. Teufel. "Columbia Multi-document Summarization: Approach and Evaluation", In Document Understanding Conf., 2001.

[3] N. Wacholder, D. Evans and J. Klavans. “Automatic Identification and Organization of Index Terms for Interactive Browsing”. In Joint Conf. on Digital Libraries ’01, 2001.
[4] A. F. R. Rahman, H. Alam and R. Hartono. “Content Extraction from HTML Documents”. In 1st Int. Workshop on Web Document Analysis (WDA2001), 2001.
[5] Suhit Gupta, Gail E. Kaiser, David Neistadt, Peter Grimm: DOM-based content extraction of HTML documents. WWW 2003: 207-214
[6] Liu L., Pu C., and Tang W. WebCQ: Detecting and delivering information changes on the web. International Conference on Information and Knowledge Management. 2000
[7] Caverlee J., Buttler D., Liu L., Discovering Objects in Dynamically Generated Web Pages. Technical Report, Georgia Institute of Technology. 2003
[8] Chang, C-H, Lui, S-L., IEPAD: Information Extraction based on pattern discovery. WWW-10, 2001
[9] C. Bouras, V. Poulopoulos, V. Tsogkas. PeRSSonal's core functionality evaluation: Enhancing text labeling through personalized summaries. Data and Knowledge Engineering Journal, Elsevier Science, 2008, Vol. 64, Issue 1, pp. 330 – 345.
[10] DOM, Document Object Model W3C standard, http://www.w3.org/DOM/
[11] HTML, Hyper-Text Markup Language, HTML 4.01 Specification, W3C standard, http://www.w3.org/TR/REC-html40/
[12] HTML Tidy Open Source Project, http://tidy.sourceforge.net/