A Fast and Accurate Approach for Main Content Extraction based on Character Encoding

Hadi Mohammadzadeh*, Thomas Gottron†, Franz Schweiggert*,and Gholamreza Nakhaeizadeh‡
*Institute of Applied Information Processing
University of Ulm, D-89069 Ulm, Germany
{hadi.mohammadzadeh, franz.schweiggert}@uni-ulm.de
†Institute for Web Science and Technologies
Universität Koblenz-Landau, D-56070 Koblenz, Germany
gottron@uni-koblenz.de
‡Gholamreza Nakhaeizadeh
Institute of Statistics, Econometrics and Mathematical Finance
University of Karlsruhe, D-76128 Karlsruhe, Germany
nakhaeizadeh@statistik.uni-karlsruhe.de

***Abstract*—This paper presents a novel approach for extracting the main content from Web documents written in languages not based on the Latin alphabet. In practice, the HTML tags are based on the English language and, certainly, the English character set is encoded in the interval [0,127] of the Unicode character set. On the other hand, many languages, such as the Arabic language, use a different interval for their characters. In the first phase of our approach, we apply this distinction for a fast separation of the Non-ASCII from the English characters. After that, we determine some areas of the HTML file with high density of the Non-ASCII character set and low density of the ASCII character set. At the end of this phase, we use this density to identify the areas which contain the main content. Finally, we feed those areas to our parser in order to extract the main content of the Web page. The proposed algorithm, called DANA, exceeds alternative approaches in terms of both, efficiency and effectiveness, and has the potential to be extended also to languages based on ASCII characters.**

***Keywords*-Main Content Extraction; Information Retrieval; UTF-8; HTML Documents; ASCII and Non-ASCII character set**

I. Introduction

Content extraction is the process of identifying the main content and/or removing the additional items [1]. With exponential growth of text based information stored in Web pages, accurate extraction of this information has become very important. This is the main reason why several researcher papers address Main Content Extraction (MCE) from Web pages. MCE can be considered as a preprocessing step for text mining and Web information retrieval. Furthermore, such main contents are very valuable as an input for many devices that have limited presentation capacity, such as mobile phones, speech readers, etc. [2]. Figure 1 shows an example of a Web page with a highlighted main content.

From a technical point of view, most of the previous approaches [1], [3] use HTML tags to separate the main content from the extraneous items. This implies the need to employ a parser for the entire Web page. Consequently, the computation costs of these MCE approaches include the overhead for the parser.

Figure 1. An example of Web pages with the selected main content.

In the early stage of the Internet, the contents of the most Web pages were written in English language. Now, especially in the last decade, a large part of information is being published also in other languages, for example Spanish, German, French, etc. Except for the non-English languages mentioned here, there are also many languages using Non-ASCII codes for their characters. The Unicode character set (UCS), which was introduced after ASCII and ISO-8859*, considers an exact interval for each language. Some of these intervals, however, have no character in common with the English character set. The DANA approach presented in this

paper exploits this fact to realize MCE for web documents in Arabic, Farsi, Pashto, and Urdu languages. By working on the binary character encoding directly, we achieve an improvement in time performance. Moreover, our approach outperforms all other MCE algorithms also in extraction performance, i.e. detects the main content more accurately and reliably.

The rest of this paper is organized as follows: Section II deals with some of the recent related works. Section III explains the UTF-8 encoding form. DANA will be explored in Section IV. In Sections V and VI we present the results, conclusions and proposals for future work.

II. Related Work

In the last decade, many scientists and researchers have been working on MCE. Several algorithms have been introduced and many papers in this field have been published. Finn et al. [4] describe the process of extracting and classifying information from HTML documents for the purpose of integrating them into digital libraries. They proposed the "Body Text Extraction" (BTE) approach, which extracts a continuous part of the HTML document as main content. Pinto et al. [5] introduced the Document Slope Curves (DSC) as an extended model of BTE. Incorporating a windowing technique, they can find more than one part of a fragmented main content in an HTML document. The Crunch framework [2] builds a DOM tree from an HTML document through an HTML parser. Then, by traversing the DOM tree, rather than raw HTML tags, and by using a number of filtering techniques, the main content of HTML Web pages is extracted. Mantratzis et al. [6] proposed another algorithm which operates on the DOM tree, too. This algorithm determines areas with a high hyperlink density within a web document. It separates these areas of navigation menus from the main content in an HTML Web page. In doing this, they examine DOM tree and assign specific scores to each hyperlink based on the location in the DOM tree. Debnath et al. [7] introduced two algorithms, FeatureExtracter and K-FeatureExtracter. These two algorithms identify the "primary content blocks" based on their features. First, they segment the Web pages into Web page blocks and, second, they separate the primary content blocks from the non-informative content blocks based on desired features. Gottron [1] proposed two new algorithms, Content Code Blurring (CCB) and Adapted Content Code Blurring (ACCB). Both of them are capable to work either on characters or tokens. By using the Gaussian blurring filter, CCB finds regions in an HTML document which contain mainly content and little code. Moreno et al. [3] presented a language independent algorithm, called Density, to extract the main content of an HTML Web page. This approach has two phases. First, they separate all contents from the HTML tags by using an HTML parser and make a density graph. Second, a region that has the highest density is determined as a main content.

III. Unicode and UTF-8 Encoding Form

The Unicode character set (UCS) is able to code all characters of different languages in the interval [U+0000, U+10FFFF]. UCS has several encoding forms, for example UTF-8, UTF-16, and UTF-32. In UTF-8, one character is saved in one to four bytes at maximum. In addition, UTF-8 has a special characteristic: it reserves the same character codes which come from ASCII codes. Thus, the first 128 characters, which include English characters, require only one byte of space, with a value guaranteed to be less than 128. Other characters which are used in other languages need two, three or four bytes. All characters of non-English-languages about which we will talk in this paper use exactly two bytes. For example, the Arabic character set is represented in the interval [U+0600, U+06FF]. The important point is that the value of each of these two bytes is greater than 128. Therefore, we are easily able to distinguish single-byte character sets, which include the English character set, from two-bytes character sets.

IV. DANA

The algorithm we present here consists of three phases:

A. First Phase: Character Set Separation

First of all, we emphasize again that all characters which are used in HTML tags, CSS, and JavaScript codes are part of the ASCII character set. On the contrary, while the main content in Arabic language documents might contain ASCII characters, e.g. for English words or special characters such as $>$, $<$, $/$, most of the characters belong to a Non-ASCII character set.

In this phase of the algorithm, our aim is to count the number of ASCII and Non-ASCII characters of each line of the HTML file[1]. We explained in detail in previous section that one byte is allocated for each ASCII character and, certainly, the value of this byte is less than 128. On the other side, Non-ASCII characters, which are used in Persian, Arabic, Pashto, and Urdu, take two bytes instead of one with values greater than 127. Now, we need only a simple condition to distinguish if a byte represents an ASCII character or not. By regarding to this condition, we are able to count the number of ASCII and Non-ASCII characters of each line of an HTML file, which can be saved in two one-dimensional arrays T1 and T2, respectively.

[1]We apply a preprocessing step to normalize line length and, thereby, render the approach independent from the actual line format of the source code.

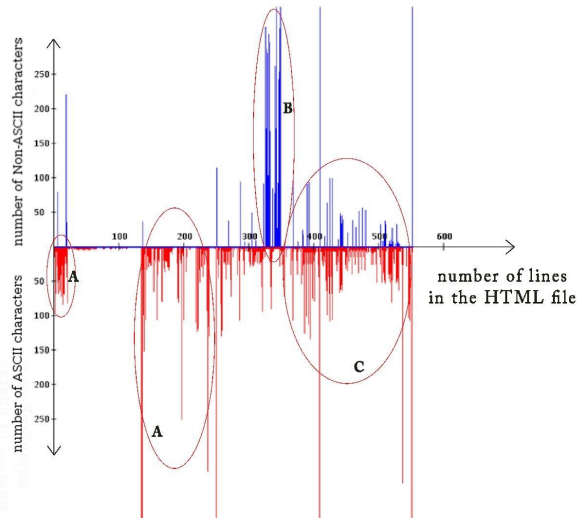

Figure 2. An example plot that shows the density of the main content and extraneous items.

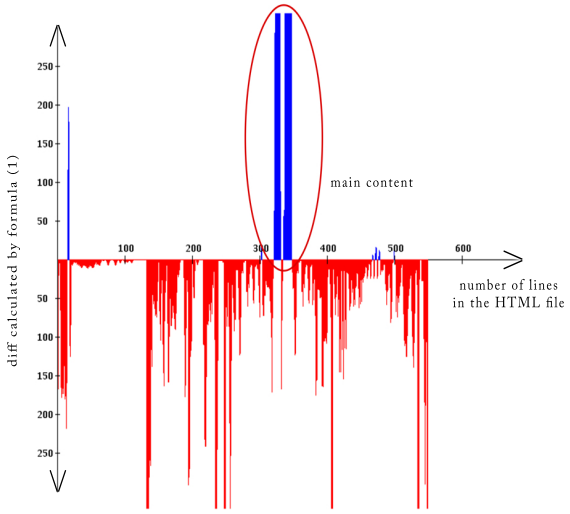

Figure 3. Smoothed version of figure 2 after using formula 1 for each column. The main content appears more clearly now.

B. Second Phase: Finding Main Content Regions

In this phase, we are looking to find areas in the HTML file with the most ASCII and the least Non-ASCII. To illustrate our approach, we depict two diagrams. In the first diagram, Figure 2, we draw two columns – one above and one below the x-axis – with the length equal to the number of Non-ASCII and ASCII characters, as stored in T1 and T2, for each line of the HTML file. For example, suppose that the i-th line of an HTML file has y1 Non-ASCII and y2 ASCII characters. Then, two lines with the length equal to y1 and y2 are drawn above and below the x-axis. Our hypothesis is, that the main content is typically located above the x-axis.

In Figure 2, the measurement unit for the x-axis is the number of lines in the HTML file. The measurement unit for the y-axis is the number of Non-ASCII (positive) and ASCII characters (negative) for each line of an HTML file.

Web site	URL	Size	Languages
BBC	http://www.bbc.co.uk/persian/	598	Farsi
Hamshahri	http://hamshahrionline.ir/	375	Farsi
Jame Jam	http://www.jamejamonline.ir/	136	Farsi
Ahram	http://www.jamejamonline.ir/	188	Arabic
Reuters	http://ara.reuters.com/	116	Arabic
Embassy of Germany, Iran	http://www.teheran.diplo.de/Vertretung/teheran/fa/Startseite.html	31	Farsi
BBC	http://www.bbc.co.uk/urdu/	234	Urdu
BBC	http://www.bbc.co.uk/pashto/	203	Pashto
BBC	http://www.bbc.co.uk/arabic/	252	Arabic
Wiki	http://fa.wikipedia.org/	33	Farsi

Table I
EVALUATION CORPUS OF 2,166 WEB PAGES.

Now, we interpret Figure 2 for finding the main content. There are three kinds of regions in Figure 2 :

- Regions that have low or near zero density of columns above the x-axis and have high density of columns below the x-axis. We observed that these regions typically consist mainly of HTML tags, JavaScript and CSS codes. We label these areas with A.
- We see only one region, B, which has a high density of columns above the x-axis and low density of columns below the x-axis. This region, certainly, contains the main content.
- There are some regions that have medium density of columns above and below the x-axis. These regions form parts of navigation menus, panels, or other related link lists. Here, normally, the density of the columns below the x-axis is somehow more than the density of the columns above the x-axis because in HTML files we need to write many tags to make menus or extraneous items. One of these areas is outlined with C.

Now, the problem of finding the main content in an HTML Web page becomes the problem of finding regions such as region B. In the next three steps we will explain how we find a region like B in an HTML file:

1) For all columns i we calculate diff_i by using formula 1. In this formula, $\mathrm{T1_i}$ and $\mathrm{T2_i}$, for example, are the number of Non-ASCII and ASCII characters of line i in an HTML file. The result is a smoothed plot that can be seen in Figure 3.

$$\begin{aligned} \mathrm{diff_i} &= \mathrm{T1_i} - \mathrm{T2_i} \\ &+ \mathrm{T1_{i+1}} - \mathrm{T2_{i+1}} \\ &+ \mathrm{T1_{i-1}} - \mathrm{T2_{i-1}} \end{aligned} \quad (1)$$

If $\mathrm{diff}_i > 0$ we draw a line with length diff_i above the x-axis. Otherwise, we draw a line with length $|\mathrm{diff}_i|$ below the x-axis. Unlike Figure 2, a large part

	Al Ahram	BBC Arabic	BBC Pashto	BBC Persian	BBC Urdu	Embassy	Hamshahri	Jame Jam	Reuters	Wikipedia
ACCB-40	0.8714	0.8255	0.8594	0.8925	0.9476	0.7837	0.8420	0.8398	0.8997	0.7364
BTE	0.8534	0.4957	0.8544	0.5895	0.9606	0.8095	0.4801	0.7906	0.8891	**0.8167**
DSC	0.8706	*0.8849*	0.8398	*0.9505*	0.8962	0.8238	*0.9482*	0.9142	0.8510	0.7471
FE	0.8086	0.0600	0.1652	0.0626	0.0023	0.0173	0.2251	0.0275	0.2408	0.2250
KFE	0.6905	0.7186	0.8349	0.7480	0.7504	0.7620	0.6777	0.7833	0.8253	0.6244
LQF-25	0.7877	0.7796	0.8436	0.8410	0.9566	0.8596	0.7650	0.7372	0.8699	0.7735
LQF-50	0.7855	0.7772	0.8374	0.8279	0.9544	0.8561	0.7673	0.7240	0.8699	0.7719
LQF-75	0.7733	0.7727	0.8374	0.8190	0.9544	0.8516	0.7560	0.7240	0.8699	0.7497
TCCB-18	*0.8861*	0.8265	*0.9121*	0.9253	0.9898	*0.8867*	0.8712	*0.9292*	*0.9593*	0.8142
TCCB-25	0.8737	0.8608	0.9091	0.9271	*0.9916*	0.8832	0.8884	0.9240	0.9583	0.8142
Density	0.8787	0.2016	0.9081	0.7415	0.9579	0.8818	0.9197	0.9063	0.9336	0.6649
DANA	**0.9845**	**0.9633**	**0.9363**	**0.9944**	**1.0**	**0.9350**	**0.9797**	**0.9452**	**0.9670**	0.6740

Table II
EVALUATION RESULTS BASED ON F1-MEASURE.

of menus and additional news in Figure 3 have been hidden.

2) Now in Figure 3, we recognize all regions above the x-axis. In this example, there are three regions, one near the y-axis and two others in the middle of the x-axis. In addition, we count the number of characters for each region and also we specify the position of regions in Cartesian coordinate. It is obvious that among all regions, the region with the maximum number of characters belongs to the main content.
3) Finally, all regions shaping the main content will be discovered. For simplicity, we define a new set $\mathrm{R} = \{\mathrm{r}_1, \mathrm{r}_2, ..., \mathrm{r}_\mathrm{n}\}$ of all regions recognized in the previous paragraph. Now, we have two states: (1) $n = 1$, (2) $n > 1$. If $n = 1$, then r_1 is the only region of the main content. Otherwise, if $n > 1$, at the beginning we find a region, $\mathrm{r}_m \in R$, with maximum number of characters. Again, we define a new empty set P, denoting the set of all regions comprising the main content at the end of this phase, and add r_m to this set. For finding all other regions of the main content, we use algorithm 1. In this algorithm, $\mathrm{d}(\mathrm{r_i}, \mathrm{r_j})$ returns distance between two regions r_i and r_j and the gap parameter is configured with a value of 20. The first and second loop, respectively, discover all regions in the left and right side of r_m comprising the main content.

C. Third phase, extracting the main content from selected regions using a parser

In final phase, we feed all HTML lines determined in the previous phase as an input to a parser [1]. Following our hypothesis, the output of the parser is exactly the main content.

V. EVALUATION

As discussed earlier we work on four languages: Arabic, Farsi, Pashto, and Urdu. To evaluate DANA, we follow the approach proposed in [8] for MCE applications.

Algorithm 1 Finding All Regions Comprising MC.

```
P = {r_m}, R = {r_1, r_2, ..., r_n}
for i = 0 → m − 2 do
    if d(r_{m−i}, r_{m−i−1}) ≤ gap then
        P = P ∪ {r_{m−i−1}}
    else
        break
    end if
end for
for i = m → n − 1 do
    if d(r_i, r_{i+1}) ≤ gap then
        P = P ∪ {r_{i+1}}
    else
        break
    end if
end for
```

A. Test Corpus and Metrics

As evaluation corpus we use 2,166 Web pages from different web sites (see table I). In order to calculate the accuracy of DANA, we first provided a manually crafted *gold standard* for the main content of all HTML files. Then, we compare the output of our algorithm with the corresponding gold standard. Hence, we need a metric to compare the *gold standard file* with the produced *cleaned file*. As in [8], [1], [9], [3], we use the Longest Common Subsequence (LCS) to find the overlap between the gold standard and the cleaned file. Now by counting the number of tokens of gold standard and cleaned files, g and m respectively, and the number k of tokens return by the LCS function we evaluate the accuracy of the algorithm by applying adoptions of the classical Information Retrieval performance measures [8]: Recall, Precision, and F1-measure, as defined in formula 2:

$$\mathrm{r} = \frac{\mathrm{length(k)}}{\mathrm{length(g)}}, \quad \mathrm{p} = \frac{\mathrm{length(k)}}{\mathrm{length(m)}}, \quad \mathrm{F1} = 2 * \frac{\mathrm{p} * \mathrm{r}}{\mathrm{p} + \mathrm{r}} \qquad (2)$$

B. Results

Table II gives statistics showing the average F1 scores of DANA and other 11 algorithms on 2,166 selected Web pages from 10 different Web sites. The bold values show the highest F1 score and the italic numbers represent the highest F1 score among all algorithms except DANA. In

addition, in table III we compute the processing time (MB/s) of DANA and other methods. By looking to these two tables, the following important points are recognized:

- As can be seen from six Web pages, Al Ahram, BBC Arabic, BBC Persian, BBC Urdu, Hamshahri, and Reuters, DANA achieves F1 score more than 0.95 and especially on BBC Urdu with an F1 score of exactly 1. No other method shows such a high effectiveness.
- In table II only BTE and only on Wikipedia web documents achieves an F1 score greater than DANA. Wikipedia documents have already been observed to be very difficult for MCE algorithms [1], [8]. By looking inside the Wikipedia HTML file, we discover that there are big gaps between the regions composing the main content. Looking at DANA's recall of 0.5734 it can be seen that it erroneously discards large parts of the main content. In the previous section, we configured the gap parameter with a value of 20. If the gap parameter is set to 160 instead of 20, then DANA achieves a recall of 0.8364, a precision of 0.8974 and an F1 score of 0.8571. In this case, DANA outperforms all other algorithms. In our outlook at further work, we will suggest some ideas how to overcome this drawback of DANA to parameterize the gap value.
- Among all eleven algorithms only DSC and TCCB achieve F1 scores close to but never as high as DANA.
- We can see that DANA also shows good efficiency, of around 19.43 MB/S. Therefore, in comparison with the comparable methods in this paper – DSC, TCCB-18 and TCCB-25, which have extraction performance close to our algorithm – DANA has an acceptable efficiency. On Wikipedia, BTE achieves extraction performance better than DANA, but DANA is about 100 times faster than BTE.

VI. Conclusions and Future Work

In this paper, we proposed a novel main content extraction algorithm, DANA, with considerable effectiveness. Results show that DANA determines the main content with previously unseen accuracy. Achieving an average F1-measure $>$ 0.935 on the test corpus used in this paper, it outperforms all previous methods. Also, DANA succeeded to achieve F1 score greater than 0.96 on over six Web sites and a perfect value of 1 on BBC Urdu.

In the future and for next research steps, we will follow several ideas: (1) Use unsupervised learning methods, such as k-Means clustering, to group several areas in an HTML file contributing to the main content of Web pages. This allows for discarding the parameter setting for gaps between main content blocks and to overcome the problem observed on Wikipedia documents. (2) Generalizing DANA by differentiating between tags and text/content to propose a new language-independent content extraction approach with higher effectiveness and accuracy.

	Time Performance (Megabyte/Second)
ACCB-40	0.40
BTE	0.17
DSC	7.76
FE	14.33
KFE	11.76
LQF-25	1.25
LQF-50	1.25
LQF-75	1.25
TCCB-18	17.09
TCCB-25	15.86
Density	7.62
DANA	19.43

Table III
Average processing time (MB/s).

References

[1] T. Gottron, "Content code blurring: A new approach to content extraction," in *DEXA '08: 19th International Workshop on Database and Expert Systems Applications*. IEEE Computer Society, Sep. 2008, pp. 29 – 33.

[2] S. Gupta, G. Kaiser, D. Neistadt, and P. Grimm, "DOM-based content extraction of HTML documents," in *WWW '03: Proceedings of the 12th International Conference on World Wide Web*. New York, NY, USA: ACM Press, 2003, pp. 207–214.

[3] J. Moreno, K. Deschacht, and M. Moens, "Language independent content extraction from web pages," in *Proceeding of the 9th Dutch-Belgian Information Retrieval Workshop*, 2009, pp. 50–55.

[4] A. Finn, N. Kushmerick, and B. Smyth, "Fact or fiction: Content classification for digital libraries," in *DELOS Workshop: Personalisation and Recommender Systems in Digital Libraries*, 2001.

[5] 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 *JCDL '02: Proceedings of the 2nd ACM/IEEE-CS joint conference on Digital libraries*. New York, NY, USA: ACM Press, 2002, pp. 46–55.

[6] C. Mantratzis, M. Orgun, and S. Cassidy, "Separating XHTML content from navigation clutter using DOM-structure block analysis," in *HYPERTEXT '05: Proceedings of the sixteenth ACM conference on Hypertext and hypermedia*. New York, NY, USA: ACM Press, 2005, pp. 145–147.

[7] S. Debnath, P. Mitra, and C. Lee Giles, "Identifying content blocks from web documents," in *Foundations of Intelligent Systems*, ser. Lecture Notes in Computer Science, 2005, pp. 285–293.

[8] T. Gottron, "Evaluating content extraction on HTML documents," in *ITA '07: Proceedings of the 2nd International Conference on Internet Technologies and Applications*, Sep. 2007, pp. 123–132.

[9] T. Gottron, "An evolutionary approach to automatically optimise web content extraction," in *IIS'09: Proceedings of the 17th International Conference Intelligent Information Systems*, 2009, pp. 331–343.