USING UTF-8 TO EXTRACT MAIN CONTENT OF RIGHT TO LEFT LANGUAGE WEB PAGES

Hadi Mohammadzadeh, Franz Schweiggert
Institute of Applied Information Processing, University of Ulm, D-89069 Ulm, Germany

Gholamreza Nakhaeizadeh
Institute of Statistics, Econometrics and Mathematical Finance, University of Karlsruhe, D-76128 Karlsruhe, Germany

Keywords: Main content extraction, Information retrieval, UTF-8, HTML documents, Right to left languages.

Abstract: In this paper, we propose a new and simple approach to extract the main content of Right to Left language web pages. Independence to DOM tree and HTML tags is one of the most important features of the proposed algorithm. In practice, HTML tags have been written in English and we know that the English character set is located in the interval [0,127]. In most languages which are written from Right-to-Left (R2L) such as the Arabic language, however, a definite interval of the Unicode character set is used that is certainly not in this interval. In the first phase of our approach, we apply this distinction to separate the R2L characters from the English ones. Then for each HTML file, we determine the density of the R2L characters and the density of Non-R2L characters. That part of the HTML file with high density of the R2L characters and low density of the Non-R2L characters contains the main content of the web page with high accuracy. The proposed algorithm has been tested, evaluated and compared with the last main content extraction approach on 2166 selected web pages.

1 INTRODUCTION

Content extraction is the process of identifying the main content (MC) and/or removing the additional items (Gottron, 2008). With exponential growth in the amount of web pages on the Internet, worthy extraction of accurate information from web pages has become very important. This is the main reason why many authors have paid attention to the main content extraction (MCE) from web pages. Such main contents are very valuable, and enable input for many devices that have limited storage capacity, such as mobile phones, speech readers, etc. Furthermore, MCE can be considered as preprocessing for text mining.

In the early stage of the Internet, most of the web pages were written in the English language. Now, especially in the last decade, a large portion of information has been published in other languages such as Spanish, German, and French. In addition to the non-English languages mentioned above, there are several other languages which are written from right-to-left (R2L). These languages have had a significant growing rate on the Internet in the last few years.

Unicode character set (UCS), which was introduced after ASCII and ISO-8859*, considers an exact interval for every language. Some of them such as the Arabic UCS, however, have no common characters with the English ones. In the literature there are several approaches addressing MCE. (Gottron, 2008) and (Moreno et al., 2009) are recent examples.

From a technical point of view, most of the above mentioned approaches use the HTML tags to separate the MC from the extraneous items. This leads to using a parser (Moreno et al., 2009) for all web pages which consequently increases the computation time. Thus, the other goal of our research is to increase the performance and accuracy of the MCE algorithms dealing with R2L languages. Figure 1 shows an example of web pages with the selected MC.

In Section 2, we review related work that provides a brief description of some MCE algorithms. In Section 3, we elaborate the UCS and the UTF-8 encoding form. The main part of this paper will be explained in Section 4, where we introduce our algorithm and its evaluation. Section 5 shows the results and compares our approach with the last main content extraction algorithm. In Section 6, we discuss our conclusion and give some suggestions for future work.

Mohammadzadeh H., Schweiggert F. and Nakhaeizadeh G..
USING UTF-8 TO EXTRACT MAIN CONTENT OF RIGHT TO LEFT LANGUAGE WEB PAGES.
DOI: 10.5220/0003508502430249
In *Proceedings of the 6th International Conference on Software and Database Technologies* (ICSOFT-2011), pages 243-249
ISBN: 978-989-8425-76-8

Figure 1: An example of web pages with the selected MC.

2 RELATED WORK

Since two decades ago, many scientists and researchers have been working on main content extraction. Several algorithms have been introduced and many papers have been published on this subject. (Finn et al., 2001) described the process of extracting and classifying information from the HTML documents for the purpose of integrating it into digital libraries. They proposed the "Body Text Extraction" (BTE) approach, which explored a continuous part of the HTML document. It also contained a high percentage of text against low percentage of tags from a web page by tokenizing the document and making a binary vector. (Pinto et al., 2002) introduced the Document Slope Curves (DSC) method, which is an extended model of the BTE and was generated through a binary vector. Long and low sloping regions of this graph represent the main content (text without HTML tags). With considering windowing technique, they could find more than one continuous part of text in an HTML document. (Gupta et al., 2003) brought a new framework and named it Crunch. This framework is making a DOM tree from an HTML document through an HTML parser. Then, by navigating DOM tree recursively, rather than using a raw HTML markup, and utilizing a number of filtering techniques, the main content of HTML web pages can be extracted. (Mantratzis et al., 2005) proposed a new algorithm whose function was based on DOM tree as well. This algorithm determines the areas with high hyperlink density within a web document, so it can separate these areas from the main content in web pages. In doing this, they examined DOM tree and assigned specific scores to each hyperlink based on location in DOM tree. (Debnath et al., 2005) 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 noninformative content blocks based on desired features. (Gottron, 2008) brought two new algorithms. The Content Code Blurring (CCB) and the Adapted Content Code Blurring (ACCB) are capable of working either on characters or tokens. CCB finds the region in an HTML document which contains mainly content and little code. In order to do this, the algorithm, by using the Gaussian blurring filter, determines a ratio of content to code for each single element in the content code vector (CCV) in the vicinity of each element and makes a new vector, named Content Code Ratio (CCR). Now a region with high CCR values contains the main content. In ACCB, all anchor-tags are ignored during the creation of the CCV. Two parameters influence the behavior of these two algorithms, so tuning these two parameters will be very important just to produce acceptable results. (Moreno et al., 2009) presented a language independent algorithm (tested on English, Italian and German languages) for the main content extraction. This approach similar to CCB has two phases. In the first step, they separate texts from the HTML tags by using an HTML parser; afterwards, the extracted texts are saved in an array of strings L. In the second step, a region in the array L that has the highest density will be determined as a main content. In addition to finding the highest density area in the array L, two parameters influence the behavior of the algorithm. The first parameter, C1, determines minimum required length for texts in each element of the array L to be selected and inserted to the new array of String R, which is considered to keep the high density region of text. The second parameter, C2, specifies the acceptable distance between lines in R and the lines which want to be added to R.

3 UNICODE AND UTF-8 ENCODING FORM

In this section, we explain Unicode character set and UTF-8 encoding form in detail.

3.1 Unicode Character Set

Before UCS was introduced, ASCII [developed to

ISO 8859*] and EBCDIC were used on computers. Thereby, only one byte was allocated for saving one character; consequently only 256 characters could be coded. By considering this limitation, rows in interval [128, 255] in the encoding table were used by different characters of different languages.

Since the introduction of UCS, where only one special number was mapped to each character, we are able to use all characters of diffrent languages on computers. At first, from 1991-1995, only 16 bits were reserved for each character, but when the new version of UCS was introduced (July 1996), it was possible to save a character in 21 bits. The newly defined UCS code all characters in the interval [U+0000, U+10FFFF]. There are several encoding forms in UCS, such as UTF-8, UTF-16, and UTF-32. In each of these encoding forms, respectively, one character can be saved in one to maximally four bytes, one or two words, or 32 bits.

3.2 UTF-8 Encoding Form

As we mentioned in 3.1, UTF-8 is a variable length encoding form in UCS. This encoding form can code all characters in UCS and has a special characteristic: it reserves the same character codes from ASCII that makes UTF-8 backward-compatible. UTF-8 represents each character in one to four bytes. Thus in UCS, the first 128 characters, which include English characters and correspond to the first 128 ASCII character set, need only one byte and have values less than 128. Other characters which are used in other languages need two, three or four bytes. All letters of non-English-languages which we will discuss in this paper take exactly 2 bytes, for example the Arabic character set has been represented in the interval [U+0600, U+06FF] and we call them R2L characters. In UTF-8, each character which needs more than one byte will be coded in such a manner that each byte of this character is greater than 127 and so it can be distinguished from one-byte characters with value less than 128. Consider the following example. Letter ب in Arabic language (b in Latin) has been defined with the value 0x0628. This code can be saved in two bytes as below. First, we convert this number to an equivalent binary value:
0x0628 = 0B0000110000101000
Second, the binary value should be divided into three parts:

000	011000	0101000

Third, two right parts will be added in following format, respectively:

11000000	10000000

UTF-8 provides this format for two bytes character sets in interval [U+0080, U+07FF]. The result is:

11011000	10101000

Now these two values are greater than 127. Therefore, we can easily separate one-byte characters with value less than 128 from double-byte characters.

4 ALGORITHM: R2L MAIN CONTENT EXTRACTION

4.1 Algorithm

The algorithm we present here consists of two steps:

First Step: Separating R2L Characters from Non-R2L Characters

In the following we define two sets, S1 and S2:
S1 = {All characters belonging to UCS R2L languages}
S2 = {All first 128 characters of UCS}
We know that English characters, which are used in HTML tags, have values less than 128 and therefore can be classified to S2. All characters of R2L languages use two bytes with a value greater than 127, and therefore they are classified to S1. This rule helps us to separate R2L language characters from the first 128 characters of UCS.

In the first step, the algorithm reads one HTML file as a stream of bytes and then by using the above rule it distinguishes whether the generated byte is a member of S1 or S2. Now, the characters in each line[1] of the file are separated into two parts: characters that are a member of S2, and the ones that are a member of S1. We defined a new structure to save these two types of characters. It should just be mentioned that sometimes we see an HTML file that has been written in one line. In our program, there is a preprocessing section which can divide a single-line-HTML file (sometimes into more than one line but still having several HTML tags on one line) as if each HTML tag places on one line.

```
struct HTML_file_line{
     String whole_line;
     String EPL;    // English Part Line
     String NEPL;   // Non English Part Line
}
```

For saving all the lines of the HTML file, we need to declare an array, T, of the above structure. In

[1]A line is a sequence of characters which is terminated by \n

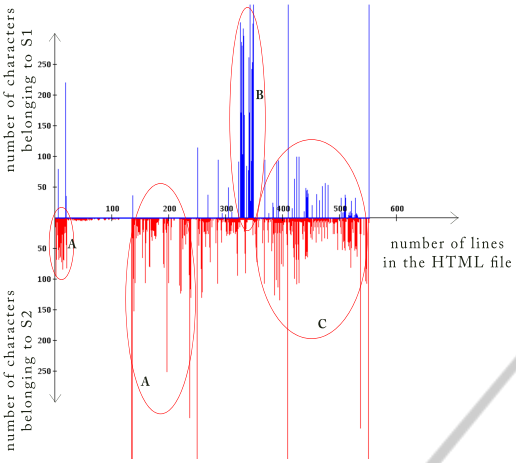

Figure 2: An example plot shows the density of the main content and extraneous items.

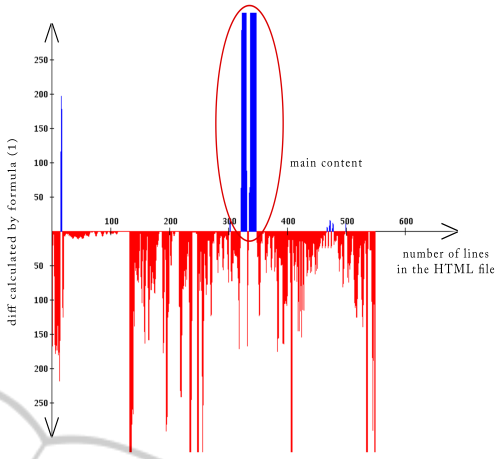

Figure 3: New generation of Figure 2 that MC can be seen easily.

the structure HTML_file_line, the first field saves all characters of a line because in the next step of the program, we need to process entire lines. For each line, second and third fields save all characters that are a member of S2 and S1, respectively.

Second step: Finding a Region Comprising MC in Array T

After saving all characters of the HTML file in the array T, we then recognize an area in the array T, where characters in fields of NEPL have high density and characters in fields of EPL have low density. Figure 2 depicts the length of strings for both EPL and NEPL fields for a sample of HTML file, where for each line of the HTML file, a vertical line is drawn on one side or on both sides of the x-axis. For the field NEPL, a vertical line with identical length is drawn upside of the x-axis and similarly, for the field EPL a vertical line with equal length is drawn downside of the x-axis. It is obvious that the main content will be found on the upside of 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 upward and downward is the number of characters which are members of S1 and S2, respectively, in each line of the HTML file.

Here we interpret Figure 2 to find MC. There are three types of regions:

- Regions that have low or near zero density of columns above the x-axis and contain a high density of columns below the x-axis. It can be observed that these regions consist of JavaScript and CSS codes, at most. We label these areas with A.
- One region has a high density of columns above the x-axis and a low density of columns below the x-axis. This region comprises the main content. This area was selected and labeled with B.
- There are some regions in which there is not too much of a difference between the density of the columns above and below the x-axis. Also, sometimes the density of the columns below the x-axis is greater than the density of the columns above the x-axis. These regions belong to menus, panels, and other additional news, and although there are some characters which are members of S1, they do not comprise the main content. One of these areas is outlined with C.

In Figure 2, two regions of A, one region of B, and one region of C have been labeled. Now the problem of finding MC in HTML web pages becomes the problem of finding a region similar to region B. In the next three steps, we find a region like B in an HTML file:

- For all columns - for example column i - we calculate the difference between the length of both columns, above and below the x-axis, and then we draw Figure 3 (see formula 1).

$$\text{diff}_i = \text{length}(\text{NEPL}_i) - \text{length}(\text{EPL}_i) \quad (1)$$

  Now if $\text{diff}_i > 0$, then we draw a line with the length diff_i above the x-axis; otherwise we draw a line with the length $|\text{diff}_i|$ below the x-axis. In Figure 3, unlike in Figure 2, a large part of menus and additional news have been removed.
- In this section, we find a column with the longest length above the x-axis. In some cases, the selected column might not be a part of MC, but belong to the menus or other extraneous items. Thus, the selected area is not really MC. To overcome this problem, we propose a solution: we extend formula (1) for columns located on the left

Table 1: Information of 2166 web pages processed by the R2L algorithm.

Web site	URL	Num. of Pages	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

and the right sides of column i, for example i-1 and i+1. Now we compute the new length of column i through formula (2). Because of the characteristics of the HTML tags, menus and extraneous items usually have long-length columns below the x-axis on the left and the right sides of the column i. This guarantees that the columns which belong to menus and extraneous items will not be selected.

$$\begin{aligned} \text{diff}_i &= \text{length}(\text{NEPL}_i) - \text{length}(\text{EPL}_i) \\ &+ \text{length}(\text{NEPL}_{i+1}) - \text{length}(\text{EPL}_{i+1}) \\ &+ \text{length}(\text{NEPL}_{i-1}) - \text{length}(\text{EPL}_{i-1}) \end{aligned} \quad (2)$$

- In this final section we are going to find the boundaries of the MC region. After recognizing the longest column above the x-axis, the algorithm moves up and down in the HTML code to find all lines belonging to the MC. But where is the end of these movements? Due to the anarchic nature of HTML codes, each HTML statement may be written in several lines, so some lines without any characters of S1 could be placed between paragraphs comprising MC. Therefore, the number of lines we need to traverse to find the next MC paragraph is defined as a parameter P. By considering this parameter, we go up or down, respectively until we can not find a line containing characters which are members of S1. At this moment, all lines we find make our MC. Based on the heuristic fitness function, which will be explained in Section 4.4, we consider a value of 8 as a default value for parameter P. This parameter is similar to the second parameter in (Moreno et al., 2009).

4.2 Data Sets

As discussed earlier, we are going to work on four languages: Arabic, Farsi, Pashto, and Urdu. To evaluate our R2L algorithm, we have collected 2166 web pages from different web sites. Table 1 explains all information about these web pages.

4.3 Evaluation

To calculate the accuracy of the R2L algorithm, we first need to manually make a *gold standard* file, which contains MC of a corresponding HTML file and is named as a *gs-file*, for each of the HTML files. Then, we need to compare the output of the R2L algorithm, which contains MC and is called *MC-file*, with a corresponding *gs-file*. Hence, we need a metric to compare the *gs-file* with the *MC-file*. In this paper, we use the Longest Common Subsequence (LCS) as a metric to compare two substrings. The LCS algorithm finds the longest common substring between two different substrings. For example, suppose that the *MC-file* and the *gs-file* contain HUMAN and CHIMPANZEE, respectively. By running LCS on these two substrings, the string HMAN is produced. Now by considering the length of the *gs-file* and *MC-file*, g and m respectively, and the length of output of the LCS function, k, we evaluate an accuracy of the R2L algorithm by applying common Information Retrieval Performance Measures - Recall, Precision, and F1-measure, (Gottron, 2007), as in formula (3):

$$\text{r} = \frac{\text{length(k)}}{\text{length(g)}},\ \text{p} = \frac{\text{length(k)}}{\text{length(m)}},\ \text{F1} = 2 * \frac{\text{p} * \text{r}}{\text{p} + \text{r}} \quad (3)$$

The F1-measure can be set with a value in the interval [0, 1]. Zero means minimum accuracy and one means maximum accuracy, which is considered as a perfect main content extraction.

Table 2: Average results for the proposed R2L algorithm based on F1-measure.

Web site	Num. of Pages	Languages	F1-measure with P = 8	Best value for Parameter P	F1-measure with new Parameter in Column 5
BBC	598	Farsi	0.9906	8	0.9906
Hamshahri	375	Farsi	0.9909	8	0.9909
Jame Jam	136	Farsi	0.9769	3	0.9872
Ahram	188	Arabic	0.9295	7	0.983
Reuters	116	Arabic	0.9356	4	0.9708
Embassy of Germany, Iran	31	Farsi	0.9536	15	0.9715
BBC	234	Urdu	0.9564	11	0.9972
BBC	203	Pashto	0.9745	8	0.9745
BBC	252	Arabic	0.987	8	0.987
Wiki	33	Farsi	0.283	16	0.3852

4.4 Fitness Function to Optimize the R2L Algorithm

All MCE algorithms try to produce better results with high accuracy, but they suffer from some parameters which should be tuned (Gottron, 2008), (Moreno et al., 2009) and (Gottron, 2009). In Section 4.1, we introduced and explained the parameter P. To find the best value for the parameter P, we implemented an heuristic fitness function (HFF). By manually searching through several web pages, we see there is, in the worst case, at most 20 lines, including empty lines and lines with HTML tags, between two paragraphs in the main content region. So we define the interval [1, 20] for HFF and then this function heuristically calculates the best value for parameter P for each special web site. All the produced results by HFF on our data sets are shown in Table 2 Column 5. Also, Column 6 shows the F1-measure that has been evaluated based on this parameter.

5 RESULTS AND COMPARISONS

5.1 Results

Table 2 contains all results, F1-measures, which have been achieved by running the proposed R2L algorithm on our gathered data sets. Column 4 shows F1-measure for P=8 where the average value of the F1-measure is calculated based on the web pages given in the Column 2. Column 6 shows the F1-measure which is calculated by using the best value for P produced by HFF. For example in Row 4 for the Ahram data set, HFF generates value 7 for P. In this case, we achieve an F1-measure of 0.983. By comparison with one in Column 4, we see a significant increase around 0.0535. However, for some data sets such as BBC Farsi, Hamshahri, BBC Pashto, and BBC Arabic, default value for the parameter P is equal to the best value. The achieved accuracy by our method is fairly high for many web sites, such as BBC, Hamshahri, Jame Jam. Our algorithm also has acceptable F1-measure for the rest web sites. An exception is the wiki web site for which our result is poor, but we can explain the reason. In the main content of the wiki web site, there are many Non-R2L characters which are eliminated automatically by our approach and lead to low accuracy. To overcome this problem, we suggest a method described in Section 6.

To summarize, if we consider P with the best value, then without any doubt, we see that our algorithm works very well, because the F1-measures for all web sites are greater than 0.97.

5.2 Comparison with other Algorithms

Due to the lack of related research for MCE from the R2L languages, we have to compare our approach with a contribution by (Moreno et al., 2009), based on the Language Independent Content Extraction Algorithm. Their approach is examined for English, Italian and German, but according to the authors, it is applicable to every language. For simplicity, we use the acronym LICEA for their contribution. Table 3 shows the results of both the R2L and LICEA approaches. We see again that in all cases except wiki, results of R2L are superior to LICEA. In some cases, such as BBC-Farsi, Embassy of Germany and BBC-Arabic, the difference between results of two approaches is relatively high. As we refer to Section 4.4, this could be because of more extraneous and space lines between paragraphs comprising main content areas of the web pages. Consequently, LICEA could not detect all main content paragraphs that would lead to low F1-measure.

Another relatively similar work is the contribution of (Gottron, 2008). The two algorithms CCB

and ACCB mentioned in Section 2 of the present paper are related to this contribution. It is interesting to notice that (Moreno et al., 2009) used the Gottron data sets and compared their approach, LICEA, with CCB and ACCB. They showed that LICEA lead to better results than CCB and ACCB. This means that if we would have compared our approach with CCB and ACCB, we would have achieved better results because as mentioned above, our results are superior to LICEA-results.

6 CONCLUSIONS AND FUTURE WORK

In this paper, we proposed a two-phase R2L main content extraction algorithm. Results show that the R2L algorithm produces satisfactory MC with F1-measure > 0.92. Our algorithm has some advantages:

- It is DOM tree and HTML-format independent; therefore, HTML statements may have some errors.
- We do not need to use parser for our algorithm. Many of the previous MCE methods have made DOM tree structure, or used HTML tags for their purpose. Using parser is time consuming.
- By fine tuning of the parameter P for each web site, we see an increase in the F1-measure.

One problem with the R2L algorithm is that it has not achieved value 1 for F1-measure, which suggests that there are words in the MCA containing Non-R2L characters. If there are only R2L characters in the main content area, then the R2L algorithm yields exactly the value 1 for the F1-measure. Otherwise, F1-measure will have a value less than 1, based on a percentage of Non-R2L characters in the main content area. To overcome this problem in the future, we have a plan to input entire lines of HTML file composing MC to an HTML parser. Then, the output of parser will be exactly the main content.

ACKNOWLEDGEMENTS

We would like to thank Dr. Norbert Heidenbluth for helping us to prepare figures and diagrams. We would like also to thank Dr. Koen Deschacht and the University of K.U.LEUVEN for providing us with the Content Extraction Software. At the end, I would like to thank Dr. Ljubow Rutzen-Loesevitz for editing this paper.

Table 3: Average results for the R2L algorithm and the LICEA reported in (Moreno et al., 2009).

Web site	R2L, P = 8	R2L, the best P	LICEA
BBC-Farsi	0.9906	0.9906	0.7755
Hamshahri	0.9909	0.9909	0.9406
Jame Jam	0.9769	0.9872	0.9085
Ahram	0.9295	0.983	0.9342
Reuters	0.9356	0.9708	0.9221
Embassy of Germany, Iran	0.9536	0.9715	0.8919
BBC-Urdu	0.9564	0.9972	0.9495
BBC-Pashto	0.9745	0.9745	0.9403
BBC-Arabic	0.987	0.987	0.302
Wiki	0.283	0.3852	0.7121

REFERENCES

Debnath, S., Mitra, P., and Giles, C. L. (2005). Identifying content blocks from web documents. In *Lecture Notes in Computer Science*, pages 285–293, NY, USA. Springer.

Finn, A., Kushmerick, N., and Smyth, B. (2001). Fact or fiction: Content classification for digital libraries. In *Proceedings of the Second DELOS Network of Excellence Workshop on Personalisation and Recommender Systems in Digital Libraries*, Dublin, Ireland.

Gottron, T. (2007). Evaluating content extraction on html documents. In *Proceedings of the 2nd International Conference on Internet Technologies and Applications*, pages 123–132, University of Wales, UK.

Gottron, T. (2008). Content code blurring: A new approach to content extraction. In *19th International Workshop on Database and Expert Systems Applications*, pages 29–33, Turin, Italy.

Gottron, T. (2009). An evolutionary approach to automatically optimize web content extraction. In *In Proceedings of the 17th International Conference Intelligent Information Systems*, pages 331–343, Krakw, Poland.

Gupta, S., Kaiser, G., Neistadt, D., and Grimm, P. (2003). Dom-based content extraction of html documents. In *Proceedings of the 12th international conference on World Wide Web*, pages 207–214, New York, USA. ACM.

Mantratzis, C., Orgun, M., and Cassidy, S. (2005). Separating xhtml content from navigation clutter using dom-structure block analysis. In *Proceedings of the Sixteenth ACM Conference on Hypertext and Hypermedia*, pages 145–147, New York, USA. ACM.

Moreno, J. A., Deschacht, K., and Moens, M.-F. (2009). Language independent content extraction from web pages. In *Proceeding of the 9th Dutch-Belgian Information Retrieval Workshop*, pages 50–55, Netherland.

Pinto, D., Branstein, M., Coleman, R., Croft, W. B., and King, M. (2002). Quasm: a system for question answering using semi-structured data. In *Proceedings of the 2nd ACM/IEEE-CS joint conference on Digital libraries*, pages 46–55, New York, USA. ACM.