Language Independent Content Extraction from Web Pages

Javier Arias Moreno
Department of computer science
Celestijnenlaan 200A
3001 Leuven, Belgium
Arias@lsi.upc.edu

Koen Deschacht
Department of computer science
Celestijnenlaan 200A
3001 Leuven, Belgium
Koen.Deschacht@cs.kuleuven.be

Marie-Francine Moens
Department of computer science
Celestijnenlaan 200A
3001 Leuven, Belgium
Sien.Moens@cs.kuleuven.be

ABSTRACT

In this paper we present a simple, robust, accurate and language-independent solution for extracting the main content of an HTML-formatted Web page and for removing additional content such as navigation menus, functional and design elements, and commercial advertisements. This method creates a text density graph of a given Web page and then selects the region of the Web page with the highest density. The results are comparable or better than state-of-the-art methods that are computationally more complex, when evaluated on a standard dataset. Accurate and efficient content extraction from Web pages is largely needed when searching or mining Web content.

General Terms

Web page cleaning

1. INTRODUCTION

When building a system for searching or mining Web content, a first task is extracting the main content and removing extraneous data such as navigation menus, functional and design elements, and commercial advertisements. Also when showing Web pages on small screens (e.g., of mobile phones) or sending text to screen readers that translate the text to a more appropriate format (e.g., text-to-speech for visually impaired people), the content extraction operation is very valuable. Content extraction (CE) is defined as the process of determining those parts of an HTML document that represent the main textual content [5]. Because different Web pages often have a different layout and a variety of configurations are possible, the task is at first sight not trivial. Recently a number of solutions have been proposed. The problem, however, is to find a solution that is generic (i.e., portable to many types of Web pages), accurate (i.e., find all important content in a precise way) and efficient (often a large number of Web pages are processed).

We designed, implemented and evaluated a content extraction system that satisfies the above requirements. Our method is simple, generic, robust and efficiently computable. The results are comparable or better than state-of-the-art methods that are computationally more complex, when evaluated on a standard dataset used in the literature. The research was done in the frame of a project where we crawl, clean, classify, summarize and index Web pages.

The remainder of the paper is organized as follows. Section 2 discusses related research. In section 3 we present our method for content extraction. Section 4 describes how we evaluate this method, and gives results and a comparison to existing methods. We conclude in section 5 where we also give some hints for future research.

2. RELATED RESEARCH

The simplest way to clean Web pages is to remove metadata and tags from the source data. The derivation is a fast, single-pass process. However, most often a deeper processing is needed in order to extract the main content, because Web data are infiltrated with advertisements and interaction menus. Early approaches to the content extraction problem heavily relied on a priori knowledge of the Web site's layout and formatting [10, 3], knowledge which could eventually automatically be learned, but the approach suggests that only a limited amount of formating templates for Web pages are used, which is an unrealistic assumption.

Gradually interest grew in building generic content extraction systems that operate on all types of Web pages. Usually the main text of a Web page is long and homogeneously formatted, while additional contents are usually highly formatted and contain little and short texts. These and other signaling cues for relevant and irrelevant content were exploited in various ways. [7] starts from the HTML tree and wraps its relevant content as a subtree that contains a large number of visible text elements, and which fans out into many children. [6] use the high ratio of link content to detect navigational menus and similar structures. [11] operate on the DOM (Document Object Model) tree, which defines the logical structure of well-formed HTML or XML documents, and identify hyperlinked clutter as text advertisements and long lists as syndicated references to other structures. [12] detect a continuous part of the document which contains text based on the analysis of so called document slope curves. A document is represented as a binary vector. HTML tags except for the ones that indicate content (e.g., font changes) are given a weight one, all other tokens are given a weight zero. From this vector a document slope curve is generated. The entries in the document slope curve graph correspond to the total of the binary vector entries up to and including each token. Long, low sloping regions of this graph represent content (text without tags). [13] use the text-to-tag ratio of lines of a document to find clusters of content in a Web page.

The closest to our approach is the Content Code Blurring (CCB) method of [5] that implements several methods to identify those parts of a Web page which contain a lot of text and few or no tags. A document is represented as a sequence of text and tag (code) characters or tokens stored as a binary content code vector. The code vector is blurred by using a Gaussian blurring filter (by iteratively spreading the values of a character or token to its neighbors until the values stabilize), after which the areas with high content bearing values are extracted. A variant of the method, Adapted Content Code Blurring (ACCB), is better suited to wiki style documents and ignores anchor tags. [5] makes a comparison with the methods described by [12] and [6], where he shows that the content code blurring method that ignores the anchor tags outperforms the former methods.

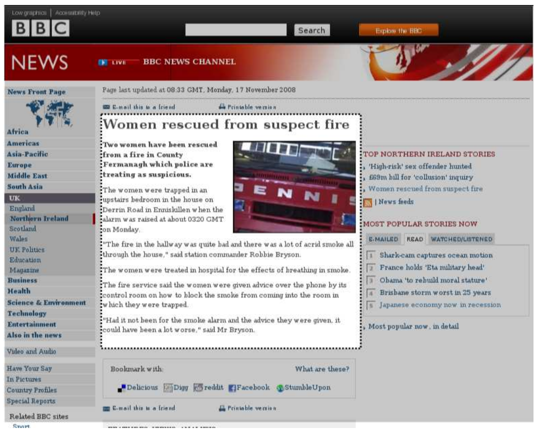
More sophisticated approaches extract the information from the visually rendered output of a Web page or other HTML content. Such an approach was following by [9], who extracted tabular data from rendered pages and [1] who classified emails based on the content that is rendered in the email browser. Although very valuable and generic, especially in an adversary setting where certain content might be present in the source, but hidden for the user of the browser, these approaches are computationally much more expensive than the method for Web page cleaning that we propose.

3. GOALS

We are given a source file containing HTML markup-tags and text. As shown in fig.1, this text consists of some relevant content ("main content") but also a lot of content that is not relevant outside the context of this particular Web page, such as navigational menus, comments, links to related articles and others. The goal of this article is to develop a method that classifies every character in the source file as being relevant or not relevant, and creates an output file that contains only the main content, cleaned from any markup-tags.

4. METHOD

We develop a method that extracts the main content from Web pages. The difficulty of this task differs largely in different settings. First, there is the setting where the Web page is known at development time. Here, the structure of the page can easily be exploited to accurately extract the main content based on simple regular expressions. In the second setting a limited number of Web sites are targeted. Here, it is possible to use automatic learning methods to creating a method that extracts the main content specifically for every Web site (i.e. site wrapping). In the third setting the Web page is unknown at development time and can originate from any Web site. This task is extremely challenging, since fully successful extraction methods need to perform a semantic analysis of both the text and struc-



Page last updated at 08:33 GMT, Monday, 17 November 2008

Women rescued from suspect fire

Two women have been rescued from a fire in County Fermanagh which police are treating as suspicious.

The women were trapped in an upstairs bedroom in the house on Derrin Road in Enniskillen when the alarm was raised at about 0320 GMT on Monday.

"The fire in the hallway was quite bad and there was a lot of acrid smoke all through the house," said station commander Robbie Bryson.

The women were treated in hospital for the effects of breathing in smoke.

The fire service said the women were given advice over the phone by its control room on how to block the smoke from coming into the room in which they were trapped.

"Had it not been for the smoke alarm and the advice they were given, it could have been a lot worse," said Mr Bryson.

- 'High-risk' sex offender hunted
- £69m bill for 'collusion' inquiry
- Women rescued from suspect fire

1. Shark-cam captures ocean motion
2. France holds 'Eta military head'
3. Obama 'to rebuild moral stature'
4. Brisbane storm worst in 25 years
5. Japanese economy now in recession

Figure 1: Example Web document with the main content marked.

ture of the Web page, which can be further complicated by common mistakes in spelling, miswritten markup-tags and an ill-defined HTML-structure. In this setting it also not possible to perform site wrapping, since the pages do not originate from a common Web site.

In this paper we propose a method that performs only a very shallow analysis of the Web page. This method does not depend on strong assumptions on the structure or content of the Web page and is fully language independent. The main idea behind our method is that a Web page has both content text (the news item, blog entry, ...) and garbage text (navigational menus, links to other articles, adverts, comments,...), but that the content texts tend to be continuous, long text with little structural markup, and that the garbage text tends to be short texts with a lot of structural markup. We make the following weak assumptions: The first assumption states that the text representing the content is separated from the garbage text with one or more markup-tags. The second assumption states that no garbage text occurs in the main content, e.g. that the main content text is continuous (not taking into account the markup-tags). The third and most important assumption states that the main content of the text contains less structural markup-tags (see below) than the garbage text.

An informal inspection of some targeted Web sites reveals that both assumption 1 and 2 are always satisfied, and these assumptions are also satisfied in our test set (see section 5.1). The third assumption, although intuitively correct, was violated in some cases. In section 5.3 we will discuss in detail when this occurred and the influence of this violation on the content extraction method.

We first locate a subset of markup-tags that modify the structure of the Web page. These tags include, but are not limited to `<p>`,`<table>`,`<BR>`,`<div>`,`<h1>`,`<h2>`,.. and `<li>`[1]. We ignore the tags that do not modify the structure of the Web page, such as `<b>`, `<a href=..>` and `<font ...>` and we also ignore data that is not content-related, such as JavaScripts, style definitions and HTML comments. We then transform the structured HTML page to a linear list

[1]see http://www.w3.org/TR/xhtml1/

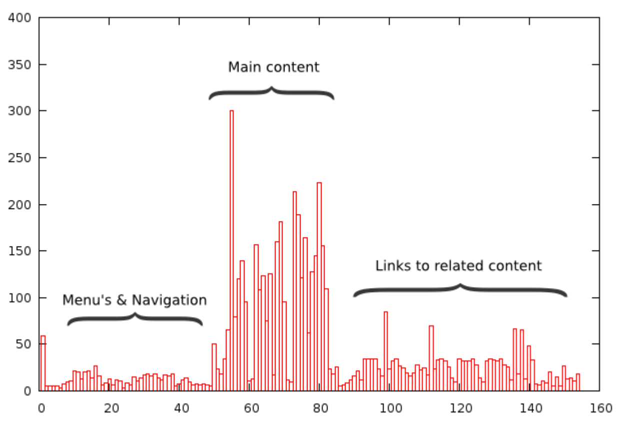


Figure 2: Example plot of the document density

of text strings $L = \{s_1, ..., s_n\}$. We parse the structure of the Web page using a robust HTML parser[2], that will, when presented with a not well-structured HTML page perform a best-effort parse. This parser visits every node in the HTML structure. If a node containing text is encountered, this text is added to the last text string in L. If a markup-tag that modifies the structure of the Web page is encountered, L is extended with one empty string. We continue this process until the entire Web page is parsed.

We build a graphical representation of the array L in fig. 2 where the x-axis represents the position of the array and the y-axis represents the length of the strings at the different positions. In a second step we analyze this graph to find the main content in the Web page. Typically, the main content for a Web page containing news articles is located in the region of L that has the highest density. We therefore convert the problem of extracting the main content of a Web page in the problem of selecting the highest density region of L, for which we have designed a simple algorithm. We first locate the string s_{max} in L with maximum length $maxL$[3]. Then a cutoff length $cutoffL$ is computed as $cutoffL = maxL * c_1$, where c_1 is a constant. We initialize the high density region R as $R = \{s_{max}\}$. We then incrementally add strings s_i to R. A string s_i is added to R iff $length(s_i) > cutoffL$ and there is a string $s_j \in R$ such that $|i - j| < c_2$, where c_2 is a constant. The algorithm terminates when no more strings can be added to R.

To create the final text $t_{automatic}$ containing the main content of the Web page, we find the leftmost string s_l in R and the rightmost string s_r in R. We then create T by concatenating all strings s_i, where i ranges from l to r (inclusive).

Optimal values for c_1 and c_2 were chosen manually when performing the experiments in section 5. The values used in this experiment were $c_1 = 0.333$ and $c_2 = 4$.

Although this algorithm is very simple, it incorporates several interesting ideas. First of all, it does not depend on the structure of any particular Web site, but uses a notion of document density which can be expected to be universal for most Web sites containing news articles. Secondly, it does not depend in any way on the text and is thus fully language independent. Thirdly, it relies only on a limited amount of the HTML-markup, thus making allowances for dirty and non-well structured Web pages.

[2]http://java.sun.com/products/archive/hotjava/

[3]Note that L and thus $maxL$ are page specific

Web site	URL	# of pages	Language
bbc	news.bbc.co.uk	1000	en
chip	www.chip.de	361	de
economist	www.economist.com	250	en
espresso	espresso.replubblica.it	139	it
golem	golem.de	1000	de
heise	www.heise.de	1000	de
manual	different	65	en,de
repubblica	www.replubblica.it	1000	it
slashdot	slashdot.org	364	en
spiegel	www.spiegel.de	1000	de
telepolis	www.telepolis.de	1000	de
wiki	de.wikipedia.org	1000	en
yahoo	news.yahoo.com	1000	en
zdf	www.heute.de	422	de

Table 1: Datasets used for evaluation, showing the name, the URL, the number of pages used in this evaluation and the languages of the datasets.

5. EXPERIMENTS

5.1 Data set used

We evaluate the proposed method on a data set previously used in state-of-the-art content extraction [5]. 14 different datasets (see table 1) were gathered from the Web. A golden standard was created for every HTML page by manually selecting the main content of every Web page. Most Web pages contain news items, although some also contain encyclopedia articles (wiki) or Web pages with different types of contents (manual). The Web sites are written in different languages : English (en), Italian (it) and German (de).

5.2 Evaluation

As described in section 3 we aim at building a method that can successfully label text in a Web page as "main content" or "garbage". In this section we describe how we evaluate the method proposed in this paper.

Although we described the task conceptually as labeling text in a Web page, in reality most (all) systems that perform this extraction task take as input the HTML source code of the Web page and return a file containing the cleaned text. The ground truth data (described above) is also stored in this cleaned text format. To evaluate the developed method, we need a metric that compares how "similar" the automatic output is compared to the manually generated output. More formally, we define t_{manual} as the main content text that was manually created and we define $t_{automatic}$ as the content text that was automatically created.

Different metrics have been proposed that measure the similarity between the two files. In this paper we use two evaluation metrics: longest common substring and longest common subsequence.

Web site	Baseline		Content extraction	
	LCString	LCSequence	LCString	LCSequence
bbc	60.16	61.52	96.32	97.17
chip	6.09	19.25	26.33	78.09
economist	30.91	66.85	45.48	91.88
espresso	69.04	77.32	82.10	89.25
golem	8.28	50.92	15.78	92.17
heise	46.57	61.47	72.28	96.82
manual	11.72	40.72	20.64	53.94
repubblica	14.77	71.95	21.57	90.74
slashdot	10.96	11.61	29.93	53.85
spiegel	8.86	55.86	13.39	86.84
telepolis	5.25	83.14	5.83	89.15
wiki	70.49	81.87	71.96	78.67
yahoo	34.73	65.75	52.36	94.58
zdf	14.39	67.50	25.13	82.93

Table 2: Average results for the baseline and automatic extraction method for LCString and LCSequence evaluation metrics, given in F1-measure (%).

The longest common substring metric (LCString) [2] finds the longest continuous string that appears both in the automatic output and the manual output. For example, the LCString of the strings "the dog jumps over the brown fox" and "the fox jumps over the brown dog" is " jumps over the brown " (of length 20). This metric is useful since it focuses on the longest continuous string, thus highly penalizing any discarded words (or punctuation marks) in the center of the text, which could possible carry high semantic value (e.g. imagine that at some point in the text the word "not" would not be extracted by the automatic method, thus possibly changing the entire meaning of a sentence). On the other hand, a major disadvantage of this method is that it treats all symbols identical, e.g. that discarding a space in the center of the text could possibly half the LCString, thus halving the score on a certain document.

The longest common subsequence metric (LCSequence) [8] finds the longest sequence of characters that appear in that order in both the automatic output and the manual output. For example, the LCSequence of the strings "the dog jumps over the brown fox" and "the fox jumps over the brown dog" is "the jumps over the brown " (of length 23). Notice how LCString is always a substring of LCSequence. This metric is less strict in that it assigns only a modest penalty to missing characters.

We have opted for two character based algorithms (in contrast to for instance [4]), since we feel that word based algorithms are harder to implement (because the characters on a Web page need to be correctly split into words which is not relevant for this task).

For both metrics we calculate, given the length of the longest common string or sequence s_{max}, the familiar information retrieval metrics of precision, recall and F1-measure, as follows:

$$precision = \frac{length(s_{max})}{length(t_{automatic})}$$

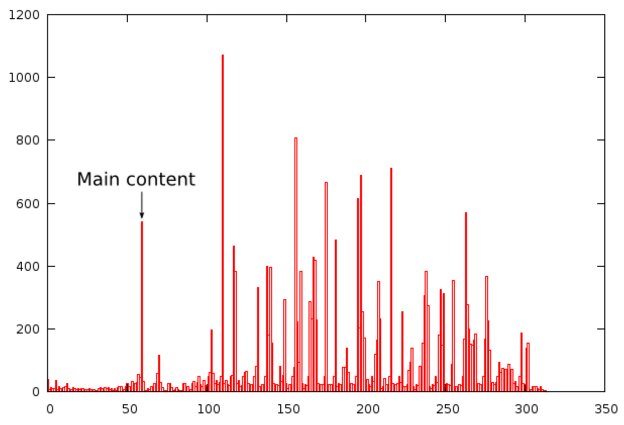


Figure 3: Document density graph for an example of the slashdot corpus.

$$recall = \frac{length(s_{max})}{length(t_{manual})}$$

$$F1 = 2 * \frac{precision * recall}{precision + recall}$$

5.3 Results

We perform a first set of experiments where we compare the raw output of the proposed method with the manual extracted texts. This method yielded results that were lower then expected, caused by superfluous spaces in the manual extracted texts. These spaces do not influence in any way the rendering of the page (apart from splitting words) and we thus feel that they can be ignored. All results reported ignore any spaces in both automatic and manual extracted texts.

We compare our approach with a baseline method that extracts *all* texts from a given Web page, removing markup information, but does not perform any content selection. Table 2 shows that the method proposed here results in a significant increase over the baseline. This can be explained by the fact that although the baseline achieves near perfect recall (since the main content will certainly be part of the extracted text), it suffers from a low precision (since it extracts all text). Our method generally also achieves a high recall, but also a high precision because of the dynamic content selection methods.

Table 2 shows that for some datasets (bbc, yahoo, heise) our method achieves near perfect F1-measure. For many other datasets (spiegel, economist, repubblica, espresso, telepolis, golem) our method achieves an F1-measure of 85% or more, which is certainly satisfying given the complexity of the task.

The proposed method achieves disappointing (although better then baseline) results on only two datasets, slashdot and manual. On the slashdot dataset we achieve a LCSequence F1-measure of 53.85%. The reason for this result can easily be explained by fig. 3. We see here that the string representing the main content (as indicated) is only a small subset of the entire text on the Web page. Closer inspection reveals that the remaining text appearing after this content

	Baseline		Content extraction	
Web site	Here	In [5]	Density	ACCB
bbc	61.52	59.5	97.17	92.4
chip	19.25	17.3	78.09	70.3
economist	66.85	61.3	91.88	89.0
espresso	77.32	62.4	89.25	87.5
golem	50.92	50.2	92.17	95.9
heise	61.47	57.5	96.82	91.6
manual	40.72	37.1	53.94	41.9
repubblica	71.95	70.4	90.74	96.8
slashdot	11.61	10.6	53.85	17.7
spiegel	55.86	54.9	86.84	86.1
telepolis	83.14	85.8	89.15	90.8
wiki	81.87	82.3	78.67	68.2
yahoo	65.75	58.2	94.58	73.2
zdf	67.50	51.4	82.93	92.9

Table 3: Average LCSequence F1-measure (in %) results of the density method reported here and the ACCB method reported in [5].

Web site	ACCB	density
bbc	1.0	0.361
chip	8.0	0.314
economist	15.0	0.294
espresso	16.0	0.317
golem	9.0	0.337
heise	12.0	0.341
manual	20.0	0.353
repubblica	14.0	0.355
slashdot	13.0	0.353
spiegel	15.0	0.351
telepolis	52.0	0.377
wiki	28.0	0.377
yahoo	13.0	0.315
zdf	1.0	0.318

Table 4: Average processing time (in s/Mb) for our density extraction method and the ACCB method reported in [5].

are comments. Indeed, a page on the slashdot Web site typically consist of a small text describing some newsworthy fact and many comments. Often the length of an individual comment is larger than the length of the news item. It is thus easy to understand how our method, relying heavily on the text density, fails for these documents.

The result on this dataset shows one of the limits of our approach. Since we assume that the density vector has a shape similar to fig. 2, we expect our method to fail on documents with a the density vector that has a very dissimilar shape. This is also the reason why the results on the wiki Web site are not excellent. We can thus see that it is important to consider the type of the Web pages before employing this technique, although in the course of the Acknowledge project we experienced that most targeted Web sites did not pose a serious problem for successful cleaning.

We found that the reason for the low performance on the manual dataset is not due to the automatic extraction method, but due to the golden standard for that dataset. For many files, the golden standard is not correct and misses large parts or all of the main content for a particular Web site. We are not aware of the reason for this errors, but note that the low result on this dataset in [5] can probably also be attributed to this incorrect golden standard.

It is hard to compare our results with existing work on this task. We are currently only aware of one work that also employs this dataset [5]. The authors evaluate their methods using the LCSequence metric, using a word based algorithm, and not a character based algorithm as reported here. Although this might lead to slightly different results, we feel that these metrics are still close enough to allow for a comparison.

Table 3 compares our content extraction method to the currently best method (Adapted Content Code Blurring) reported in [5]. We also compare the baseline results reported here with our baseline result. Surprisingly, we notice that for some datasets (e.g. espresso, respectively 62.4% and 77.32%) there is a large difference between the two baseline systems, although for other systems this difference is only very modest (e.g. golem, with respectively 50.2% and 50.92%). This difference can only partially be explained for by different evaluation metrics, but must also be caused by a weaker baseline in [5], possibly caused by a less robust HTML parser or by accidentally adding non-textual content (e.g. JavaScripts) to the baseline result.

When comparing our content extract method to the ACCB method, we see that the proposed density method achieves on average 84.00% F1-measure where the ACCB method achieves 78.16%. We achieve significantly higher results for 9 datasets (bbc, chip, economist, espresso, heise, manual, slashdot, wiki and yahoo), comparable results for 1 dataset (spiegel) and significantly lower results for 4 datasets (golem, repubblica, telepolis and zdf). These results might indicate that our method achieves generally better results. This belief is further strengthened by the fact that our method seems more robust, since our method achieves less then 75% F1-measure on only two (slashdot and manual) dataset, compared to 5 datasets (chip, manual, slashdot, wiki and yahoo) in [5].

We feel that the better results are largely due to the method of creating the density vector employed here. Our choice to use only structural tags, and ignore other mark-up tags makes that this vector reflects closer the real structure of the page, and that text elements that are structurally closer together are also closer together in this vector.

Table 4 shows the processing times needed for the different dataset on a 1.66Ghz Intel cpu. We also compare our processing times with the times reported in [5]. This comparison might not be very accurate since we are not aware of the speed of the computer that generated these results, but still indicates that the proposed method performs at least comparable and probably faster then this state-of-the-art method.

6. CONCLUSIONS

We have presented a novel method for content extraction from Web pages, sometimes also referred to as Web page cleaning. This method relies on a single heuristic that the main content of a HTML page has a high density of text characters and low density of structural code. We have shown that this method performs comparable to, or better than state-of-the-art methods. Furthermore, it has the following valuable properties : (1) it is simple, and easy to implement, (2) it is fast, processing up to 3.4Mb of HTML code per second, (3) it runs robustly on dirty or not well-formed HTML code and (4) it does not use the content of the text itself and is thus language-independent

Furthermore, we have proposed to make a distinction between structural and non-structural markup-tags. A comparison with another state-of-the-art method has shown that making this distinction improves results and allows for more robust methods.

Although we have shown that text density is an important heuristic when extracting content from Web pages, it is naive to expect that all Web pages can be successfully cleaned using this heuristic alone. Therefore, in the future more powerful methods will have to be developed. We think that several research directions could prove to be promising: Firstly, methods that perform an analysis of an entire Web site (as compared to a single Web page) could discover the common structure and texts of all pages of a certain Web site. It can be expected that this common structure and texts do not belong to the main content for a particular Web site. Secondly, one could perform an analysis of the text in a Web page, and learn that certain words do (e.g. “written by”, “author”) or do not (e.g. “close this window”, “comments”) belong to the main content.

The method as described here works on the raw density values. It might be advantageous however to create a more abstract representation which potentially allows more powerful algorithms. For instance, we could approximate a smooth function such as a polynomial function to the density values using a least squares method. We could then use the maximum, minimum, first and second order derivatives to select the highest density region.

Acknowledgments

We would like to thank Thomas Gottron for providing us the dataset which was used in the evaluation of this work. The work reported in this paper was partially financed by the Acknowledge project (IBBT) and partially by the AMASS++ project (IWT 060051).

7. REFERENCES

[1] A. Bergholz, G. Paass, F. Reichartz, S. Strobel, M.-F. Moens, and B. Witten. Detecting known and new salting tricks in unwanted emails. In *CEAS 2008: Proceedings of the Fifth Conference on Email and Anti-Spam*, 2008.

[2] P. E. Black. ”longest common substring”. In *Dictionary of Algorithms and Data Structures [online].* http://www.nist.gov/dads/, 2004, retrieved 17/11/2008.

[3] 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.

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

[5] T. Gottron. Content code blurring: A new approach to content extraction. *International Workshop on Database and Expert Systems Applications*, 0:29–33, 2008.

[6] 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*, pages 207–214, New York, NY, USA, 2003. ACM.

[7] W. Han, D. Buttler, and C. Pu. Wrapping web data into xml. *SIGMOD Record*, 30(3):33–38, 2001.

[8] D. S. Hirschberg. Algorithms for the longest common subsequence problem. *J. ACM*, 24(4):664–675, 1977.

[9] B. Krüpl, M. Herzog, and W. Gatterbauer. Using visual cues for extraction of tabular data from arbitrary HTML documents. In *WWW '05: Special interest tracks and posters of the 14th international conference on World Wide Web*, pages 1000–1001, New York, NY, USA, 2005. ACM.

[10] S.-H. Lin and J.-M. Ho. Discovering informative content blocks from web documents. In *KDD '02: Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 588–593, New York, NY, USA, 2002. ACM.

[11] 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*, pages 145–147, New York, NY, USA, 2005. ACM.

[12] 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*, pages 46–55, New York, NY, USA, 2002. ACM.

[13] T. Weninger and W. H. Hsu. Text extraction from the web via text-to-tag ratio. *Database and Expert Systems Applications, International Workshop on*, 0:23–28, 2008.