An Efficient Language-Independent Method to Extract Content from News Webpages

Eduardo Cardoso
ecardoso@inf.puc-rio.br

Iam Jabour
ijabour@inf.puc-rio.br

Eduardo Laber
laber@inf.puc-rio.br

Departamento de Informática, PUC-Rio
Mq. de São Vicente 225, RDC, Rio de Janeiro, RJ, Brazil

Rogério Rodrigues
roger@microsoft.com

Microsoft Corporation
Avenida Rio Branco 1, 1611,
Rio de Janeiro, RJ, Brazil

Pedro Cardoso
pedrolazera@gmail.com

Departamento de Informática,
PUC-Rio
Mq. de São Vicente 225,
RDC, Rio de Janeiro, RJ,
Brazil

ABSTRACT

We tackle the task of news webpage segmentation, specifically identifying the news title, publication date and story body. While there are very good results in the literature, most of them rely on webpage rendering, which is a very time-consuming step. We focus on scenarios with a high volume of documents, where performance is a must. The chosen approach extends our previous work in the area, combining structural properties with hints of visual presentation styles, computed with a quicker method than regular rendering, and machine learning algorithms. In our experiments, we took special attention to some aspects that are often overlooked in the literature, such as processing time and the generalization of the extraction results for unseen domains. Our approach has shown to be about an order of magnitude faster than an equivalent full rendering alternative while retaining a good quality of extraction.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Information filtering*; I.7.m [**Document and Text Processing**]: Miscellaneous; I.2.6 [**Artificial Intelligence**]: Learning—*Concept learning*

General Terms

Algorithms, Experimentation, Performance

Keywords

News segmentation, webpage rendering

DocEng'11, September 19–22, 2011, Mountain View, California, USA.

1. INTRODUCTION

We face an ever growing amount of content being produced every day. In this scenario, identifying and extracting the contents of a webpage, discarding templates and similar non-relevant parts of the page, is useful for several applications. To name a few, screen reading software for the visually impaired may focus on the content and skip templates and other irrelevant content; search engines can more cleanly store the page's data, which provides more accurate search results; and small screen devices, such as modern mobile phones, can use it to increase readability.

The task of news segmentation consists in identifying the key regions of the webpage. These regions might have a smaller or bigger role depending on the application. For our purposes, we consider the title, publication date and story body as regions of interest. Very good results have already been reported for this task: using the F-score metric, described in section 3.5, we observe works reaching 97% of F1 for title detection [16, 18], 87% of F1 for publication date detection [18] and 94% of F1 for news body detection [18] for an exact DOM node metric. We believe that these results are satistactory for some applications and there's not room for large improvements other than refining current approaches. However, most approaches rely on rendering the web page, which demands lots of processing and, consequently, time.

Our focus is on large-scale document processing, specifically the page processing step of search engine's web crawlers, extending the previous work of [8]. Search engines keep local versions of the webpages for indexing purposes. With a cleaner copy of a page, search results tend to be much more relevant as terms that would otherwise be part of the page, but not part of the relevant content (templates, advertisements, etc.), are discarded. In addition, the knowledge of the news title and its publication date can be useful for ranking results, either by relevance (since the title is a general summary of the text) or by date. Some works in the literature have shown how information extraction can improve document retrieval, such as [6] and [28].

For this specific scenario, we feel that there is a lack of suitable approaches. Rendering webpages is not an option

because it is a time-consuming task, as shown by [8, 10], and it would slow down the throughput of these systems. Thus, we propose a new approach that more closely keeps up with the high volume of documents in this scenario, while still producing satisfactory results. First, our method locates a DOM node that includes the page's relevant content in its subtree. Then, we proceed to remove noise from this subtree. Finally, we use machine learning models that identify the title and publication date. For these models, we use structural features and visual presentation information computed by a simplified CSS parser. The reduced subset of the page in which we apply these models, along with the simplified CSS implementation, provides us with the necessary performance we were looking for.

To test our approach, we constructed a corpus consisting of 200 news webpages in English, Portuguese and Spanish. A total of 20 websites were crawled and each contributed with exactly 10 pages. These pages have been manually annotated for news title, publication date and article text. We have carefully observed the often overlooked aspects of processing time and results quality when applying our model to a website not previously seen in training, as well as those already seen. Our approach has shown to be about an order of magnitude faster than an equivalent full rendering alternative while retaining a good quality of extraction. The results we obtained in a cross-validation for seen websites using this corpus, measured with a text-based metric, are 92% of F1 for title detection, 84% of F1 for date detection and 88% of F1 for body detection. For unseen websites, the results are 91% of F1 for title detection, 77% of F1 for date detection and 88% of F1 for body detection. Slightly better results are achieved when testing against other corpora such as the NEWS600 [17], which consists of 604 pages from 177 distinct domains, all written in English.

2. RELATED WORK

Related works in the area may be classified in various ways. It is common to differentiate by their scope, which creates the notions of *site-level* and *page-level approaches*, and by the requirements to solve the problem, which range from *strictly structural* properties to a *full rendering* of the page, which provides the geometric positioning of elements and allows the use of computer vision algorithms.

Site-level approaches require some mass of example data to build a model or rules that are specific for its pages. As it is tailored for a specific group, the results are generally better, but come at the cost of high maintenance, high setup costs and limited usability due to the wrappers that are created to exploit particularities of each site's design. A good example of this approach is [22], which identifies site templates using tree edit distance.

On the opposite side of the spectrum lies the page-level approaches. These are devised to work on virtually any webpage, including those from websites never seen before. Its generality comes at the cost of slightly worse results, but requires little maintenance, has low setup costs and broad usability since the approach works independently of the site's design. Examples of these are [6, 28], which identify titles in generic webpages and [23, 24], which train a model from a single website and apply it to 11 others to extract the news article content.

The structural approaches depend on features directly extracted from the HTML file, which may or may not be converted into a DOM tree. Information such as number of nodes, link density, word tokens, among others, are considered structural features. Methods that make good use of them include [11], which describes one of the winning approaches used in the CleanEval shared task [2], and [15], which uses a token-based local classifier to identify the boundaries of article text.

The rendering approaches may include all features available structurally, but have access to other information such as geometric positioning, bounding box size, font size and font color of various elements in the webpage, commonly available in web browsers. Works making use of this information generally achieve better results than strictly structual ones, from which we may cite [16, 17, 18], which perform segmentation of news pages' content in several classes, and [10], which takes the approach of [15] and applies it to visual features with good results.

3. OUR APPROACH

In this work, we design a page-level approach that lies in between the strictly structural vs. full rendering spectrum. It uses structural properties of news web pages and visual presentation information from cascading style sheets, such as font size and color, which may be calculated without a full rendering of the web page, providing us with good execution times. With this information available, we train a machine learning model to classify DOM nodes and employ some post-processing afterwards. Also, we strive to keep it language independent. Since we don't rely on textual cues, we can apply our approach to virtually any document and observe little variation in the results obtained.

An outline of our execution pipeline is displayed in Figure 1.

3.1 Document Object Model (DOM)

The Document Object Model (DOM) [13] is a standard created by the World Wide Web Consortium (W3C) [27] to represent HTML, XHTML and XML documents such as web pages.

A document is seen as a rooted tree where each tag or element represents an *element node*. Text contained in these elements is placed in *text nodes*, which are leaves of this tree.

Other types of nodes exist, but won't have any particular impact on our approach. We will be focusing on the text nodes, as that is where the content lies.

3.2 Cascading Style Sheets (CSS)

Cascading Style Sheets [1] is a language used to describe presentation attributes of HTML and XML documents, such as font face, color and size; absolute and relative positioning; etc. It allows the separation of content and presentation in a web page and the use of several style sheet definitions to be combined in a predictable (or cascade) manner.

It was created in 1996, with the latest revisions dating from 2008. Over the years, it has seen wide adoption by most web sites as the browsers started providing support for it and presentational elements were deprecated from the HTML specification [7, 21]. All current major browsers have a complete implementation of CSS Level 1 and a near-complete implementation of CSS Level 2.1 [3, 14, 20, 26].

Simplified CSS parser: We have developed a simplified CSS parser that exempts us from using a rendering engine

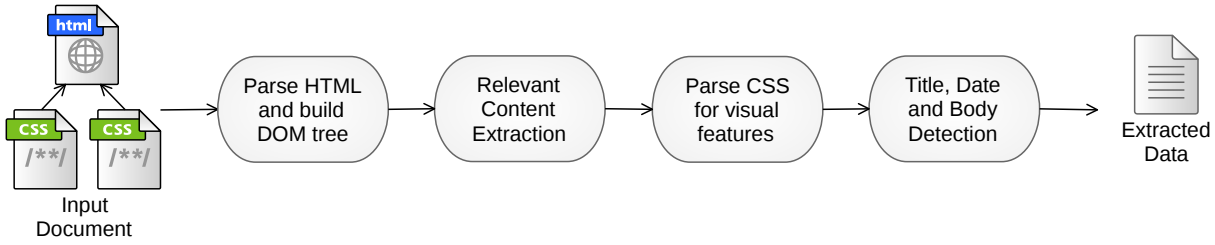

Figure 1: Outline of our execution pipeline

such as Mozilla Firefox's Gecko [4], Microsoft Internet Explorer's MSHTML [12] or Apple Safari and Google Chrome's WebKit [25]. These engines are capable of a much more accurate parsing, but end up computing much more information than we require, raising execution time. With our parser we have access to all information we need much faster, making our approach ideal for use in time-sensitive tasks such as crawling the web. A time comparison between rendering engines and our approach is made in the next section.

3.3 Attributes

For each text node, we use 9 attributes computed from the parsed web page as features for our machine learning models. These are:

1. the text length, measured in characters;
2. amount of digit characters present in the text;
3. percentage of digit characters in the text;
4. font size, normalized to pixels;
5. font size relative to the biggest font size seen, measured in percentage;
6. whether the node is presented in bold letters;
7. the amount of similarly styled nodes in the document;
8. a measure of distance between the current node and the document's node that contains the title;
9. edit distance between the node text and the document's `<title>` tag contents.

The first six attributes are very natural. The last three, however, deserve some further explanation.

Attribute 7 attempts to capture the uniqueness of a presentation style. This is motivated by the observation that the title is often easily recognized by humans because it stands out from the rest of the text in the page, usually because of a combination of style attributes. Two presentation styles are considered similar if the font size, font color, font family (serif, sans-serif font, etc.) and bold text attributes are the same.

Attribute 8 is discarded for title detection. It is only used for date detection, measuring the distance to the detected title node, as described in more detail in section 3.4.2. Similarly, for date detection, attribute 9 is not used.

Attribute 9 measures edit distance as

$$EditDistance(node\ text,\ title\ tag)\ /\ length(title\ tag)$$

where $EditDistance(\cdot, \cdot)$ is the classic Levenshtein metric [9] with weights adjusted to penalize deletions from the node text much more than insertions. This is done to smooth out the impact of the web site identification that usually comes in the document title while penalizing the loss of the node's information.

3.4 Extraction process

3.4.1 Extraction of relevant content

The first step in our approach is the extraction of relevant content from the news page. We define as *relevant* the body of the news story, its title and associated metadata, such as author and publication date. That is, the relevant content is what is left after removing all templates associated with the webpage.

To detect the relevant content, we employ the NCE algorithm described in [8]. This algorithm is quite fast and provides a very good starting point for the news body, which will later be improved by the next steps of our approach. It is based on the assumption that a *separator node* exists in the webpage; defined as a node for which the textual content of the subtree rooted on it contains most or all of the relevant content of the page. We proceed by refining the selection of relevant content with the identification and removal of subtrees that present a high link density or repeated textual patterns. The former is indicative of navigational links (menus), advertisements, related stories, etc.; and the latter often indicates a comments section at the end of the news body. As a definition of link density we use the number of characters in `<a>` tags in a subtree over the number of characters in the textual content of that subtree. The remaining subtrees combined should provide high values of precision and recall for the relevant content. We suggest the reader to refer to [8] for more details on the implementation and the different steps of the algorithm.

3.4.2 Title and date detection

We apply a binary classification model at each step, separating the title from all the rest, then the date from all the rest. We use an ensemble of decision tree models for this task [19]. For this, we used the Weka tool [5] for training and model settings were kept with default values, with no specific optimizations.

First, we classify the document title. Second, we classify the publication date. Our premise is that dates are generally close to the document title [24], so we use the classified title as a feature for the date classification. For each node, we calculate its distance to the classified title and use that to

guide our model. This constitutes attribute 8, mentioned in section 3.3.

It's worth considering what happens with the date classification model when no title node is detected. We make some considerations on this regard in the next section.

3.4.3 Post-processing

Each of the previously mentioned steps receive some post-processing.

For titles, we only allow one node per document to be detected. In case two or more are detected (excluding the `<title>` tag), we always pick the one that comes first on the document. However, if no title nodes are detected, we take the `<title>` tag as the title.

For dates, we cluster all detected nodes by their proximity in the document and select the cluster closest to the detected title. In case no title was detected (and thus the `<title>` tag is used), no clustering takes place and all nodes are considered. Next, nodes which would add large amounts of text are discarded. This is usually a sign of noise being added as our observations show that publication dates are well separated from the document's content.

Finally, we redefine as body every non-title and non-date relevant node visited in a depth-first search that starts at the detected title and ends at the last node from the extracted relevant content.

3.5 Metrics

We have been using two different metrics to assess our results: the first is node-based and the second uses bag of words. A discussion on the advantages and disadvantages of each of them is provided in the following subsections.

The final user application should dictate which metric to use. In this paper, unless otherwise noted, all results reported use the bag of words metric.

For evaluating both methods, we use the concepts of *precision*, *recall* and *F-measure*. A high value for precision indicates few wrong classifications, while a high value for recall indicates most of the annotated data is retrieved. They are calculated for each class (title, date, etc.) as follows:

$$\text{precision} = \frac{\#correctly\ classified\ elements}{\#total\ classified\ elements}$$

$$\text{recall} = \frac{\#correctly\ classified\ elements}{\#total\ elements\ in\ class}$$

It can be tricky to improve the results on these metrics: to improve recall, more elements may be classified; however, the more elements classified, the smaller the precision obtained.

As these two measures are often reported together, it is common to use the harmonic mean of both measurements, called F-measure or F-score. Here, we will be working with the *F1-score*, which means precision and recall are given the same weight. To calculate the F1-score, we use the following formula:

$$\text{F1-score} = \frac{2 \times precision \times recall}{precision + recall}$$

3.5.1 Node-based

Since we are dealing with a DOM tree, it is natural and convenient to think of a node-based metric. However, this may not always be the best solution.

One issue we found is that all nodes have the same weight when calculating precision and recall, regardless of their content. This means that small nodes with little information are as valuable as large nodes with lots of information. While they could be weighted by character size or word count, small nodes might contain crucial information such as the publication date or the news title. Consider the case of news titles: detecting a node that does not represent the exact news title would be a miss, but its contents might highly resemble the title, which would make it valuable in a search engine setting. Likewise for dates: detecting a node that only partially contains the date, for instance the day and month of publication, but not the exact time, would have a high impact on the precision and recall metrics, but may do no harm for a given application that disregards such time information.

3.5.2 Bag of Words

We employ a metric based on bag of words for measuring the results of our extraction. In this approach, text is represented as a collection of unordered tokens (referred to as words) which may appear repeatedly. Words are often defined as a contiguous sequence of either non-whitespace characters or characters from a given set (for instance, letters and numbers, but no punctuation). Neither approach for tokenization requires linguistic information for this task, which does not invalidate the language independence of our method for languages that make use of the Latin alphabet. Also, special care has been taken to reduce the impacts of eventual encoding errors in the text.

We prefer the bag of words approach because we have found that it more accurately represents what a human would consider as a correct or incorrect extraction. The issues present on the node-based approach are not completely absent, but smoothed out in a way that they cause less negative impact for an imperfect match.

4. EXPERIMENTS

We constructed the RCD4, a corpora of 200 news pages that we employed in our experiments. Its pages were manually annotated for news title, publication date and story body. The pages in the corpus are written in English, Portuguese and Spanish. All objects referred by the pages are also downloaded, such as images, style sheets, script files, etc., in order to keep our copies as close as possible to the originals.

Special care has been taken to collect the same amount of pages from each domain: a total of 20 websites were used, and 10 pages were downloaded from each of them. This allows us to perform balanced cross-validation tests to evaluate how well our models generalize the problem for unseen domains and how results can be improved by adding examples of pages from the same domains. We denote by *extra-site cross-validation* an experiment that trains a model on a different set of domains from the testing documents; that is, folds are domain-disjoint. Similarly, we denote by *intra-site cross-validation* an experiment that trains and tests on pages from the same domain; that is, folds are representative of the domain distribution in the corpus.

As previously mentioned, we start by extracting the relevant content of the webpage with the NCE algorithm described in [8], using it as a base for the next steps. This

reduces the amount of nodes that we need to classify in order to find the news title and publication date.

Our experiments show that a 35 to 40% speedup in total execution time is obtained by discarding the nodes that are not considered part of the relevant content. These discarded nodes have affected the quality of our results in at most a 1% decrease in F1.

Title detection: We conducted a series of experiments to evaluate the impact of visual presentation attributes. As a baseline for comparison, we use a classifier that always determines the document's `<title>` tag as the document title. The results for strictly structural features and our combined approach can be seen in Tables 1 and 2.

Method	Prec.	Recall	F1
Baseline (`<title>` tag)	0.61	0.89	0.72
3-fold CV (extra-site)	0.72	0.90	0.80
3-fold CV (intra-site)	0.73	0.91	0.81

Table 1: Title extraction results on the RCD4 corpus for strictly structural attributes

Method	Prec.	Recall	F1
Baseline (`<title>` tag)	0.61	0.89	0.72
3-fold CV (extra-site)	0.88	0.95	0.91
3-fold CV (intra-site)	0.90	0.94	0.92

Table 2: Title extraction results on the RCD4 corpus for both structural and visual attributes

We observe that the use of visual presentation attributes significantly improves our results, with over 10% increase in F1.

A time comparison of these approaches is given in Table 3, measured relatively to the structural approach. As an example, the value 1.77 for the Structural + Visual method indicates that it is 77% slower than the strictly structural approach. The table also includes timings for the approach that skips the relevant content detection step, thus classifying every node in the DOM tree, which we identify as "whole tree", and time estimates for an equivalent approach that uses full rendering instead of our simplified CSS parser. The rendering engine used was WebKit [25], with scripts and plugins disabled. We then proceeded to add to the rendering time of the pages the average time it took for our algorithm to run once every feature is computed. We toggled the use of images, as they might be of interest to preserve the appearance of the webpage in case geometric positioning of nodes is needed.

Date detection: Our model for date detection is title-dependent. That is, it depends on a correct classification of the title because this information is used for attribute 8 (see section 3.3) during date detection. However, when a title node is not classified, this dependency will most likely prevent our models from obtaining a correct classification.

We then employed two different models, depending on the title detection outcome. If some node is classified as title in the previous step, we proceed with the title-dependent model for dates. Otherwise, a title-independent model is used. Results for the conditional and title-dependent approaches are shown, respectively, in Tables 4 and 5.

Method	Time taken
Baseline (`<title>` tag)	0.40
Strictly structural	1.00
Structural + Visual	1.77
Structural + Visual, whole tree	3.01
WebKit rendering, no images	10.80
WebKit rendering	39.16

Table 3: Title extraction execution times on the RCD4 corpus, relative to strictly structural approach

Method	Prec.	Recall	F1
3-fold CV (extra-site)	0.88	0.66	0.75
3-fold CV (intra-site)	0.88	0.82	0.85

Table 4: Date extraction results on the RCD4 corpus for the conditional approach with two models, after post-processing

Surprisingly, we observe that the title-dependent model still seems to be the best choice as it produces more balanced results for precision and recall, specially for the extra-site cross-validation.

Body detection: For body detection, we consider every node returned from the NCE algorithm of [8] used at the start of the pipeline, excluding only those that were detected as title or publication date. As only a few nodes were excluded from the returned set, the results were unaffected, as shown in Table 6.

We experimented with a post-processing stage that would also consider all nodes between the title and the first detected body node, but the results weren't very revealing: very little have changed, and often times for the worse.

4.1 Results comparison

For the sake of comparison, we have applied our methods to the NEWS600 corpus [17], which consists of 604 pages from 177 distinct domains, all written in English. Despite the small size of our corpus, the results obtained in NEWS600 are consistent with the numbers obtained in our experiments, which reinforces the stability of our results. The results obtained are shown in Table 7, along with the best published results we have found for the same corpus.

It is worth noting that we require considerably less computational resources. Our small set of less than a dozen features behaves quite well considering that listed results tend to use tens of thousands of features and depend on a full rendering for geometric positioning.

Method	Prec.	Recall	F1
3-fold CV (extra-site)	0.79	0.75	0.77
3-fold CV (intra-site)	0.85	0.83	0.84

Table 5: Date extraction results on the RCD4 corpus for the title-dependent model, after post-processing

Method	Prec.	Recall	F1
NCE of [8]	0.82	0.92	0.87
After our pipeline	0.82	0.92	0.87

Table 6: Body extraction results on the RCD4 corpus

Approach	Label	Prec.	Recall	F1
Baseline (`<title>`)	Title	0.63	0.93	0.75
Our approach	Title	0.90	0.96	0.93
SVM of [16]	Title	0.99	0.96	0.97
Our approach	Date	0.87	0.85	0.86
CRF of [18]	Date	0.83	0.92	0.87
Our approach	Body	0.82	0.95	0.88
SVM of [16]	Body	0.90	0.98	0.94

Table 7: Results on the NEWS600 corpus, ordered by F1. The baseline and our results are measured using bag of words.

5. CONCLUSION

We presented an approach that makes use of some visual features of webpages while being approximately an order of magnitude faster than a full rendering approach.

We applied it to the task of segmenting a news webpage, obtaining satisfactory results with a small number of features and regardless of the language of the story and website where it has been published. In our experiments, we achieve at least 91% of F1 for title detection, 77% of F1 for date detection and 88% of F1 for body detection.

Future work: We would like to extend our approach to other relevant metadata in the news domain, such as author and news agency, when available, and other tasks and domains in the web where visual features might be useful, such as e-commerce and price comparison.

6. REFERENCES

[1] Çelik, T., Bos, B., Hickson, I., and Lie, H. W. Cascading style sheets level 2 revision 1 (CSS 2.1) specification. Candidate recommendation, W3C, Sept. 2009. http://www.w3.org/TR/2009/CR-CSS2-20090908.

[2] Cleaneval home page. http://cleaneval.sigwac.org.uk. [Online; accessed 18-January-2011].

[3] DOM CSS Properties – MDC Doc Center. https://developer.mozilla.org/en/DOM/CSS. [Online; accessed 13-April-2011].

[4] Gecko. https://developer.mozilla.org/en/Gecko. [Online; accessed 01-September-2010].

[5] Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., and Witten, I. The weka data mining software: an update. *ACM SIGKDD Explorations Newsletter 11*, 1 (2009), 10–18.

[6] Hu, Y., Xin, G., Song, R., Hu, G., Shi, S., Cao, Y., and Li, H. Title extraction from bodies of html documents and its application to web page retrieval. In *SIGIR '05: Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval* (New York, NY, USA, 2005), ACM, pp. 250–257.

[7] Jacobs, I., Raggett, D., and Hors, A. L. HTML 4.01 specification. W3C recommendation, W3C, Dec. 1999. http://www.w3.org/TR/1999/REC-html401-19991224.

[8] Laber, E. S., de Souza, C. P., Jabour, I. V., de Amorim, E. C. F., Cardoso, E. T., Rentería, R. P., Tinoco, L. C., and Valentim, C. D. A fast and simple method for extracting relevant content from news webpages. In *CIKM* (2009), D. W.-L. Cheung, I.-Y. Song, W. W. Chu, X. Hu, and J. J. Lin, Eds., ACM, pp. 1685–1688.

[9] Levenshtein, V. I. Binary codes with correction of deletions, insertions and substitution of symbols. *Doklady Akademii Nauk SSSR 163*, 4 (1965), 845–848.

[10] Luo, P., Fan, J., Liu, S., Lin, F., Xiong, Y., and Liu, J. Web article extraction for web printing: a dom+visual based approach. In *Proceedings of the 9th ACM symposium on Document engineering* (2009), ACM, pp. 66–69.

[11] Marek, M., Pecina, P., and Spousta, M. Web page cleaning with conditional random fields. *Cahiers du Cental 5* (2007), 1.

[12] MSHTML. http://msdn.microsoft.com/en-us/library/bb508516(v=VS.85).aspx. [Online; accessed 01-September-2010].

[13] Nicol, G., Champion, M., Hégaret, P. L., Robie, J., Wood, L., Hors, A. L., and Byrne, S. Document object model (DOM) level 3 core specification. W3C recommendation, W3C, Apr. 2004. http://www.w3.org/TR/2004/REC-DOM-Level-3-Core-20040407.

[14] Opera Presto 2.1 – Web standards supported by Opera's core – Dev.Opera. http://dev.opera.com/articles/view/presto-2-1-web-standards-supported-by/. [Online; accessed 13-April-2011].

[15] Pasternack, J., and Roth, D. Extracting article text from the web with maximum subsequence segmentation. In *Proceedings of the 18th international conference on World wide web* (2009), ACM, pp. 971–980.

[16] Spengler, A., Bordes, A., and Gallinari, P. A comparison of discriminative classifiers for web news content extraction. In *Proceedings of RIAO 2010, 9th Int. Conf. on Adaptivity, Personalization and Fusion of Heterogeneous Information* (2010).

[17] Spengler, A., and Gallinari, P. Learning to Extract Content from News Webpages. In *Proceedings of the 2009 International Conference on Advanced Information Networking and Applications Workshops* (2009), IEEE Computer Society, pp. 709–714.

[18] Spengler, A., and Gallinari, P. Document structure meets page layout: loopy random fields for web news content extraction. In *Proceedings of the 10th ACM symposium on Document engineering* (2010), ACM, pp. 151–160.

[19] Tan, P.-N., Steinbach, M., and Kumar, V. *Introduction to Data Mining*. Addison-Wesley, 2005.

[20] The WebKit Open Source Project – CSS (Cascading Style Sheets). http://www.webkit.org/projects/css/index.html. [Online; accessed 13-April-2011].

[21] van Kesteren, A. HTML 5 differences from HTML 4. W3C working draft, W3C, Aug. 2009. http://www.w3.org/TR/2009/WD-html5-diff-20090825/.

[22] Vieira, K., da Silva, A. S., Pinto, N., de Moura, E. S., Cavalcanti, J. M. B., and Freire, J. A fast and robust method for web page template detection and removal. In *CIKM '06: Proceedings of the 15th ACM international conference on Information and knowledge management* (New York, NY, USA, 2006), ACM, pp. 258–267.

[23] Wang, J., Chen, C., Wang, C., Pei, J., Bu, J., Guan, Z., and Zhang, W. V. Can we learn a template-independent wrapper for news article extraction from a single training site? In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining* (2009), ACM, pp. 1345–1354.

[24] Wang, J., He, X., Wang, C., Pei, J., Bu, J., Chen, C., Guan, Z., and Lu, G. News article extraction with template-independent wrapper. In *Proceedings of the 18th international conference on World wide web* (2009), ACM, pp. 1085–1086.

[25] WebKit. http://webkit.org/. [Online; accessed 01-September-2010].

[26] Wikipedia. Comparison of layout engines (cascading style sheets) — Wikipedia, the free encyclopedia, 2010. [Online; accessed 22-September-2010].

[27] World Wide Web Consortium (W3C). http://www.w3c.org/. [Online; accessed 14-September-2010].

[28] Xue, Y., Hu, Y., Xin, G., Song, R., Shi, S., Cao, Y., Lin, C.-Y., and Li, H. Web page title extraction and its application. *Information Processing and Management 43*, 5 (2007), 1332–1347.