Web2Text: Deep Structured Boilerplate Removal

Thijs Vogels, Octavian-Eugen Ganea and Carsten Eickhoff

Department of Computer Science,
ETH Zurich,
Switzerland
firstname.lastname@inf.ethz.ch

Abstract. Web pages are a valuable source of information for many natural language processing and information retrieval tasks. Extracting the main content from those documents is essential for the performance of derived applications. To address this issue, we introduce a novel model that performs sequence labeling to collectively classify all text elements in an HTML page as either boilerplate or main content. Our method uses convolutional networks on top of DOM tree features to learn unary classification potentials for each block of text on the page and pairwise potentials for each pair of neighboring text blocks. We find the the most likely labeling according to these potentials using the Viterbi algorithm. The proposed method improves page cleaning performance on the CleanEval benchmark compared to the state-of-the-art. As a component of information retrieval pipelines it improves retrieval performance on the ClueWeb12 collection.

1 Introduction

Modern methods in natural language processing and information retrieval are heavily dependent on large collections of text. The World Wide Web is an inexhaustible source of content for such applications. However, a common problem is that Web pages include not only main content, but also ads, link lists, navigation, previews of other articles, banners, *etc.* This boilerplate/template content has often been shown to have negative effects on the performance of derived applications [16,26].

The task of separating main text in a Web page from the remaining content is known in the literature as "boilerplate removal", "Web page segmentation" or "content extraction". Established popular methods for this problem use rule-based or machine learning algorithms. The most successful approaches first perform a splitting of an input Web page into text blocks, followed by a binary labeling of each block as either main content or boilerplate.

In this paper, we propose a neural network model for boilerplate removal. In addition to traditional features, our method leverages the representational power of convolutional networks to learn unary and pairwise classification potentials for the sequence of text blocks in a page based on complex non-linear combinations of DOM based features. We find the labeling that maximizes the joint label

sequence probability using the Viterbi algorithm. The effectiveness of our method is demonstrated on a number of standard benchmarking datasets.

The remainder of this document is structured as follows. Section 2 gives an overview of related work. Section 3 formally defines the main-content extraction problem, introduces the block segmentation procedure and details our model. Section 4 empirically demonstrates the merit of our method on several benchmark datasets for content extraction and document retrieval.

2 Related Work

Early approaches to HTML boilerplate removal use a range of heuristics and rule-based methods. Finn *et al.* [7] design an effective system called *Body Text Extractor* (BTE). Their method relies on the observation that the main content contains longer paragraphs of uninterrupted text, where HTML tags occur less frequently compared to the rest of the Web page. Looking at the cumulative distribution of tags as a function of the position in the document, Finn *et al.* identify a flat region in the middle of this distribution graph to be the main content of the page. While simple, their algorithm has two drawbacks: (1) it only makes use of the location of HTML tags and not of their structure, thus losing potentially valuable information, and (2) it can only identify one continuous stretch of main content which is unrealistic for a considerable percentage of modern Web pages.

To address these issues, several other algorithms have been designed to operate on DOM trees, thus leveraging the semantics of the HTML structure [12,20,6]. The problem with these early methods is that they make intensive use of the fact that pages used to be partitioned into sections by `<table>` tags, which is not a valid assumption anymore nowadays.

In the next line of work, the DOM structure is used to jointly process multiple pages from the same domain, relying on their structural similarities. This approach was pioneered by Yi *et al.* [26] and was improved by various others [15,25]. These methods are very suitable for detecting template content that is present in all pages of a website, but have poor performance on websites that use a single template across the whole site. In this paper we focus on single-page content extraction without exploiting the context of other pages from the same site.

Gottron *et al.* [11] propose *Document Slope Curves* and *Content Code Blurring* methods that are able to identify multiple disconnected content regions. The latter method parses the HTML source code as a vector of 1's, representing pieces of text, and 0's, representing tags. This vector is then smoothed iteratively, such that eventually it finds active regions where text dominates (content) and inactive regions where tags dominate (boilerplate). This idea of smoothing is adopted by others [21], some of which extend the concept to also deal with the DOM structure [4,24]. Chakrabarti *et al.* [3] assign a likelihood of being content to each leaf of the DOM tree while using isotonic smoothing to combine the likelihoods of neighbors with the same parents. In a similar direction, Sun *et al.* [24] use

both the tag/text ratio and DOM tree information to propagate *DensitySums* through the tree.

Machine learning methods offer a convenient way to combine various indicators of "contentness", automatically weighting hand-crafted features according to their relative importance. The FIASCO system by Bauer *et al.* [2] uses Support Vector Machines (SVM) to classify HTML blocks. These blocks are generated through a DOM-based segmentation of the page and are represented by linguistic, structural and visual features. Similar works of Kohlschütter *et al.* [16,18] employ decision trees and SVMs to independently classify blocks. Neunerdt *et al.* [22] extend this approach by reformulating the classification problem as a case of sequence labeling where all blocks are jointly tagged. They use conditional random fields to take advantage of correlations between the labels of neighboring content blocks. A similar approach [23] proved to be the most successful in the CleanEval competition [1].

In this paper, we propose a more effective set of block features that capture information from adjacent neighbors in the DOM tree. Additionally, we employ a deep learning framework to automatically learn non-linear interactions between these features as well as among neighboring text blocks, giving the model an advantage over traditional linear approaches. Finally, we jointly optimize the labels for the whole web page according to local potentials predicted by the neural networks.

3 Web2Text

Boilerplate removal is defined as the problem of labeling sections of the text of a Web page as *main content* or *boilerplate* (anything else) [1]. In the following, we discuss the various steps of our method. The complete pipeline is also illustrated in Figure 1.

3.1 Preprocessing

We expect raw Web page input to be written in (X)HTML markup. Each document is parsed as a Document Object Model tree (DOM tree) using Jsoup [13]. The first step of our algorithm is to preprocess the DOM tree by:

- Removing empty nodes or nodes containing only whitespace.
- Removing nodes that do not have any content we can extract: *e.g.* `<br>`, `<checkbox>`, `<head>`, `<hr>`, `<iframe>`, `<img>`, `<input>`.

We make use of DOM tree structure through the notion of parents and grandparents. In a raw DOM tree, however, these relationships are not always meaningful. Figure 2 shows a typical fragment of a DOM tree where two neighboring links share the same semantic parent (`<ul>`) but not the same DOM parent. To improve the semantics of features based on tree relationships such as "the number of children of a node's parent", we recursively merge nodes with exactly one child with the child. We call the resulting tree-structure the *Collapsed DOM* (CDOM).

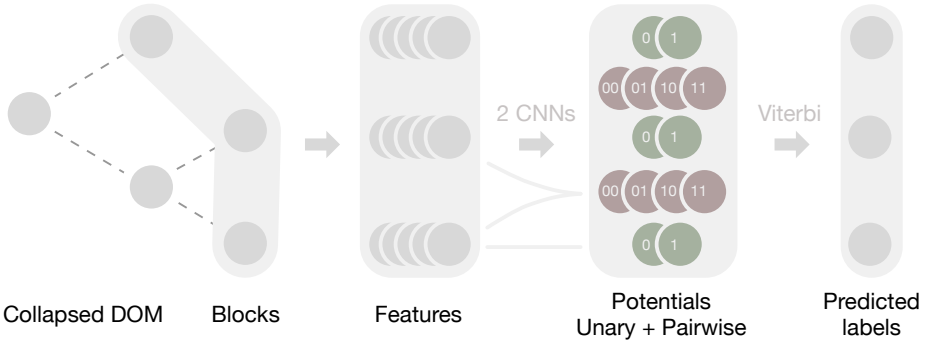

Fig. 1. The Web2Text pipeline. The leaves of the Collapsed DOM tree of a Web page form an ordered sequence of blocks to be labeled. For each block, we extract a number of DOM tree based features. Two convolutional networks operating on this feature representation yield two respective sets of potentials: unary potentials $p_i(\text{content})$, $p_i(\text{boilerplate})$ for each block i and pairwise potentials $p_{ij}(\text{content} \rightarrow \text{content})$, $p_{ij}(\text{content} \rightarrow \text{boilerplate})$, $p_{ij}(\text{boilerplate} \rightarrow \text{content})$, $p_{ij}(\text{boilerplate} \rightarrow \text{boilerplate})$ for each pair of neighboring blocks (i, j). Using the Viterbi algorithm, we find an optimal labeling that maximizes the total sequence probability as predicted by the neural networks.

3.2 Block Segmentation

Our content extraction algorithm is based on sequence labeling. A Web page is treated as a sequence of blocks that are labeled *main content* or *boilerplate*. There are multiple ways to split a Web page into blocks, ranging from letter-by-letter to paragraph-by-paragraph. The most popular methods currently used are:

i) **Lines in the HTML file**. In many HTML files, new sections will start on a new line in the file, therefore some authors use the lines of the source code as their splitting criterion. Unfortunately this does not work on minified Web pages that are stripped of white space or pages that make excessive use of newlines.

ii) **DOM leaves**. Sections on a page that require different labels are usually separated by at least one HTML tag. Therefore, it is safe to consider DOM leaves (`#text` nodes) as the blocks of our sequence. The disadvantage of this approach is that a hyperlink in a text paragraph can receive a different label than its neighboring text.

iii) **Block-level DOM leaves**. A mixed approach would be to only use block-level DOM nodes (`<p>`, `<div>`, `<table>`, `<td>`, `<br>` *etc.*) for splitting. These are tags that start (by default) on a new line in the formatted page in most browsers. The issue with this method is that, with *CSS*, block-level nodes can be restyled as inline. Therefore, this separation can be dangerous. Moreover, sometimes it makes sense to remove links from a paragraph.

We opt for using the most flexible *DOM leaves* strategy. Under this scheme, an empirical evaluation shows no cases where parts of a paragraph are wrongly labeled *boilerplate* while the rest of the paragraph is marked *main content*.

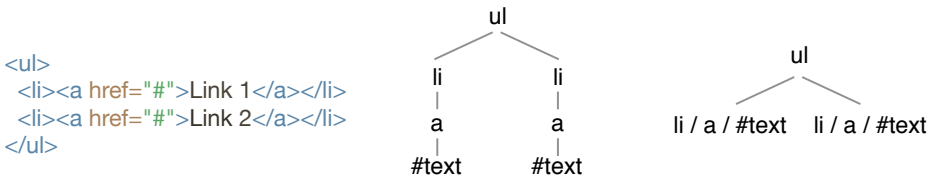

Fig. 2. Collapsed DOM: To improve the expressiveness of tree-based features, we recursively merge DOM nodes with exactly one child with their child into a single grouped node. *Left*: HTML source code, *middle*: the corresponding DOM tree, *right*: the corresponding Collapsed DOM.

Table 1. Overview of extracted node features.

logNCharacters	log(number of characters in the node)
rWords	number of words / words on the page
nSentences	number of sentences in the node
rPunctuation	number of characters $\in \{,,?,;,:,!\}$ / number of characters in node
rDashes	number of characters in $\in \{-,\_,/,\backslash\}$ / number of characters in node
rPeriods	number of periods / number of characters in node
rCharsInLink	number of characters in anchor tags / number of characters in node
endsWithPunctuation	1 if the node ends with punctuation, 0 otherwise
endsWithQuestionMark	1 if the node ends with a question mark 0 otherwise
rWordsWithCapital	ratio of the words in the node starting with a capital letter
rStopwords	ratio of the words that are in our English stop-word list
avgWordLength	average word length in the node
startRel	start position of the node in the source code / source length
endRel	end position of the node in the source code / source length

3.3 Feature Extraction

Features are properties of a node that may be indicative of it being content or boilerplate. Such features can be based on the node's text, CDOM structure or a combination thereof. Examples are the number of characters in the node, number of punctuation marks, ratio of HTML tags to text in the node string, *etc.* Table 1 gives a complete overview of the features we collect for any CDOM node. Some of them are inspired by features used in other works such as [23].

The Web page content to be labeled is a sequence of text blocks, each corresponding to a CDOM node. For each such block, we collect *block features*, consisting of the features extracted for the blocks CDOM node, the parent CDOM node and the grandparent CDOM node.

Furthermore, we capture interactions between adjacent blocks using *tree features*. These allow the learning algorithm to retain some of the tree structure of the original CDOM while operating on a linear sequence of blocks. Some examples use the following binary features:

- Blocks have the same parent / grandparent / grand-grandparent
- Blocks have the same tag / class name

– Blocks have the same tag path (*e.g.* `body>div>ul>li.navitem>a>#text`)

3.4 CNN Unary and Pairwise Potentials

We use convolutional neural networks to assign unary potentials to each text block to be labeled and pairwise potentials to each pair of neighboring text blocks. The unary potentials $p_i(\text{content})$, $p_i(\text{boilerplate})$ are the probability that a text block i is content or boilerplate, respectively. The two potentials sum to one. The pairwise potentials $p_{ij}(\text{content} \rightarrow \text{content})$, $p_{ij}(\text{content} \rightarrow \text{boilerplate})$, $p_{ij}(\text{boilerplate} \rightarrow \text{content})$, $p_{ij}(\text{boilerplate} \rightarrow \text{boilerplate})$ are probabilities that between a pair of neighboring text blocks (i, j) there is a transition from one label (content/boilerplate) to another. The pairwise potentials also sum to one for each text block pair.

For both sets of potentials, we employ convolutional neural networks with identical architectures. We use 7 layers, ReLU non-linearity, dropout with rate 0.2, a convolution filter size of 3 in each layer, and a convolution "depth" of 50 per layer, except for the last CNN layer where 10 filters are used. The CNNs receive the same input – a sequence of feature representations of the sequence of text blocks to be labeled – and output potentials for each block. The outputs for the unary network are 2 values per block that are normalized using softmax regression. The outputs for the pairwise network are 4 values per block that are normalized in the same way. This model is implemented in TensorFlow and our source code is available at [1].

3.5 Inference

The joint prediction of the most likely sequence of labels given an input Web page works as follows. We denote the sequence of text blocks on the page as $(b_0, b_1, \ldots, b_n)$ and write the probability of a corresponding labeling $(\ell_0, \ell_1, \ldots, \ell_n) \in \{0, 1\}^n$ being the correct one as

$$p(\ell_0, \ldots, \ell_n) = p_0(\ell_0) \cdots p_n(\ell_n) \cdot p_{01}(\ell_0 \rightarrow \ell_1) \cdots p_{(n-1)n}(\ell_{n-1} \rightarrow \ell_n).$$

This expression can be maximized using the Viterbi algorithm [8] to find the optimal labeling given the predicted CNN potentials.

4 Experiments

Our experiments are grouped in two stages. We begin by assessing Web2Text's performance at boilerplate removal on a high-quality manually annotated corpus of Web pages. In a second step, we turn towards a much larger collection and investigate how improved content extraction results in superior information retrieval quality. Both experiments clearly highlight the benefits of Web2Text over state-of-the-art alternatives.

[1] https://github.com/dalab/web2text

Table 2. Boilerplate removal performance comparison.

Method	Acc	Precision	Recall	F_1
CRF [23]	0.82	0.88	0.81	0.84
BTE [7]	0.75	0.76	0.84	0.80
default-ext [17]	0.79	0.89	0.74	0.81
article-ext [17]	0.67	0.89	0.50	0.64
largest-ext [17]	0.59	**0.93***	0.33	0.48
Unfluff [9]	0.68	0.90	0.51	0.65
Web2Text	**0.88***	0.89	**0.91***	**0.90***

4.1 Training Data

CleanEval 2007 [1] is the largest publicly available training set for this task. It contains approximately 1000 manually labeled Web pages of training data and a predefined held-out test set. However, this corpus has the disadvantage of providing only pairs of Web pages and corresponding cleaned extracted text (manually annotated), without the true block labeling. To recover this annotation needed for our algorithm, we use dynamic programming to align the clean text with the original Web page blocks. This method uses a simple heuristic. Random parts of the cleaned text are checked for uniqueness in the original page. If such a unique match is found, then it can be used to divide both the cleaned text and the original Web page in two parts on which the same matching method can be applied recursively.

4.2 Boilerplate Removal Performance

The first stage of our empirical performance evaluation relies on the test set of the CleanEval competition [1] and intrinsically compares the boilerplate removal performance of Web2Text to a wide range of methods described in the literature or deployed in popular libraries. Table 2 shows the results of this experiment. Statistical significance of performance differences between the proposed models and all baselines is determined using a Wilcoxon signed-rank test with $\alpha < 0.05$ and is denoted by an asterisk. We can note that Web2Text matches state-of-the-art precision while greatly excelling in terms of recall, giving it a significant overall performance margin in terms of F_1 scores as compared to popular baselines including previous CleanEval winners. Note that these numbers are obtained by evaluating each method using the same block segmentation; the *DOM leaves* strategy described in Section 3.2.

4.3 Impact on Retrieval Performance

Besides the previously presented intrinsic evaluation of text extraction accuracy, we are interested in the performance gains that other derived tasks experience when operating on the output of boilerplate removal systems of varying quality. To this end, our extrinsic evaluation studies the task of *ad hoc* document retrieval.

It is conceivable that search engines that index high-quality output of text extraction systems should be better able to answer a given user-formulated query than systems indexing raw HTML or naïvely cleaned content. Our experiments are based on the well-known ClueWeb12 collection of Web pages.[2] It is organized in two well-defined document sets, the full CW12-A corpus of 733M organic Web documents (27.3 TB of uncompressed text) as well as the smaller, randomly sampled subset CW12-B of 52M documents (1.95 TB of uncompressed text). The collection is indexed using the Indri search engine and retrieval runs are conducted using two state-of-the-art probabilistic retrieval models; The query likelihood model [14] (QL) as well as a relevance-based language model [19] (RM). Our 50 test queries alongside their relevance judgments originate from the 2013 edition of the TREC Web Track [5].

Table 3 highlights the performance of each combination of retrieval model and collection when indexing either raw or cleaned Web content. Within each combination, statistical significance of performance differences between raw and cleaned HTML content is denoted by an asterisk. Models that significantly outperform all other text extraction methods are indicated by †. We can note that, in general, retrieval systems indexing CW12-A deliver stronger results than those operating only on the CW12-B subset. Due to the random sampling process, many potentially relevant documents are missing from this smaller collection. Similarly, across all comparable settings, the query likelihood model (QL) performs significantly better than the relevance model (RM). As hypothesised earlier, text extraction can influence the quality of subsequent document retrieval. We note that low-recall methods (BTE, article-ext, largest-ext, Unfluff) cause losses in retrieval performance, as relevant pieces of content are incorrectly removed as boilerplate. At the same time, the most accurate models (CRF, Web2Text) were able to introduce considerable improvements across all metrics. Web2Text, in particular, outperformed all baselines at significance level 0.05.

5 Conclusion

This paper presents a novel algorithm for main content extraction from Web pages. The method combines the virtues of popular sequence labeling approaches such as CRFs [10] with deep learning methods that leverage the DOM structure as a source of information. Our experimental evaluation on CleanEval benchmarking data shows significant performance gains over all state-of-the-art methods. In a second set of experiments, we demonstrate how highly accurate boilerplate removal can significantly increase the performance of derived tasks such as ad hoc retrieval.

[2] http://lemurproject.org/clueweb12/

Table 3. The effect of boilerplate removal on *ad hoc* retrieval performance.

Collection	Ret. Model	Method	P@10	R@10	F_1@10	MAP	nDCG
CW12-A	QL	raw content	0.316	0.056	0.095	0.137	0.459
CW12-A	QL	CRF [23]	0.342*	0.068*	0.113*	0.147*	0.543*
CW12-A	QL	BTE [7]	0.301	0.048	0.083	0.128	0.435
CW12-A	QL	default-ext [17]	0.318	0.055	0.094	0.138	0.462
CW12-A	QL	article-ext [17]	0.298	0.049	0.084	0.126	0.433
CW12-A	QL	largest-ext [17]	0.279	0.044	0.076	0.112	0.417
CW12-A	QL	Unfluff [9]	0.304	0.051	0.087	0.128	0.428
CW12-A	QL	Web2Text	**0.361*†**	**0.079*†**	**0.130*†**	**0.154*†**	**0.578*†**
CW12-A	RM	raw content	0.278	0.048	0.082	0.121	0.439
CW12-A	RM	CRF [23]	0.301*	0.057*	0.096*	0.138*	0.487*
CW12-A	RM	BTE [7]	0.262	0.041	0.071	0.110	0.409
CW12-A	RM	default-ext [17]	0.277	0.048	0.082	0.123	0.442
CW12-A	RM	article-ext [17]	0.260	0.039	0.068	0.109	0.411
CW12-A	RM	largest-ext [17]	0.248	0.032	0.057	0.097	0.401
CW12-A	RM	Unfluff [9]	0.264	0.041	0.071	0.111	0.407
CW12-A	RM	Web2Text	**0.325*†**	**0.069*†**	**0.114*†**	**0.145*†**	**0.525*†**
CW12-B	QL	raw content	0.210	0.025	0.045	0.037	0.134
CW12-B	QL	CRF [23]	0.241*	0.031*	0.055*	0.048*	0.165*
CW12-B	QL	BTE [7]	0.193	0.019	0.035	0.030	0.121
CW12-B	QL	default-ext [17]	0.212	0.026	0.046	0.038	0.132
CW12-B	QL	article-ext [17]	0.199	0.017	0.031	0.031	0.120
CW12-B	QL	largest-ext [17]	0.178	0.015	0.028	0.024	0.107
CW12-B	QL	Unfluff [9]	0.195	0.020	0.036	0.029	0.121
CW12-B	QL	Web2Text	**0.266*†**	**0.038*†**	**0.067*†**	**0.055*†**	**0.181*†**
CW12-B	RM	raw content	0.172	0.021	0.037	0.030	0.122
CW12-B	RM	CRF [23]	0.198*	0.028*	0.049*	0.041*	0.143*
CW12-B	RM	BTE [7]	0.158	0.015	0.027	0.022	0.111
CW12-B	RM	default-ext [17]	0.170	0.020	0.036	0.029	0.124
CW12-B	RM	article-ext [17]	0.156	0.015	0.027	0.019	0.109
CW12-B	RM	largest-ext [17]	0.145	0.013	0.024	0.015	0.095
CW12-B	RM	Unfluff [9]	0.159	0.016	0.029	0.021	0.112
CW12-B	RM	Web2Text	**0.213*†**	**0.032*†**	**0.056*†**	**0.046*†**	**0.165*†**

References

1. Marco Baroni, Francis Chantree, Adam Kilgarriff, and Serge Sharoff. CleanEval: a competition for cleaning web pages. In *LREC*, 2008.
2. Daniel Bauer, Judith Degen, Xiaoye Deng, Priska Herger, Jan Gasthaus, Eugenie Giesbrecht, Lina Jansen, Christin Kalina, Thorben Krüger, Robert Märtin, Martin Schmidt, Simon Scholler, Johannes Steger, Egon Stemle, and Stefan Evert. FIASCO: Filtering the internet by automatic subtree classification, osnabruck. In *Building and Exploring Web Corpora: Proceedings of the 3rd Web as Corpus Workshop, incorporating CleanEval*, volume 4, pages 111–121, 2007.
3. Deepayan Chakrabarti, Ravi Kumar, and Kunal Punera. Page-level template detection via isotonic smoothing. In *Proceedings of the 16th international conference on World Wide Web*, pages 61–70. ACM, 2007.
4. Deepayan Chakrabarti, Ravi Kumar, and Kunal Punera. A graph-theoretic approach to webpage segmentation. In *Proceedings of the 17th international conference on World Wide Web*, pages 377–386. ACM, 2008.
5. Kevyn Collins-Thompson, Paul Bennett, Fernando Diaz, Charlie Clarke, and Ellen Voorhees. Overview of the TREC 2013 web track. In *Proceedings of the 22nd Text Retrieval Conference (TREC'13)*, 2013.
6. Sandip Debnath, Prasenjit Mitra, Nirmal Pal, and C Lee Giles. Automatic identification of informative sections of web pages. *IEEE transactions on knowledge and data engineering*, 17(9):1233–1246, 2005.
7. Aidan Finn, Nicholas Kushmerick, and Barry Smyth. Fact or fiction: Content classification for digital libraries. *Unrefereed*, 2001.
8. G David Forney. The Viterbi algorithm. *Proceedings of the IEEE*, 61(3):268–278, 1973.
9. Adam Geitgey. Unfluff – an automatic web page content extractor for node.js!, 2014.
10. John Gibson, Ben Wellner, and Susan Lubar. Adaptive web-page content identification. In *Proceedings of the 9th annual ACM international workshop on Web information and data management*, pages 105–112. ACM, 2007.
11. Thomas Gottron. Content code blurring: A new approach to content extraction. In *Database and Expert Systems Application, 2008. DEXA'08. 19th International Workshop on*, pages 29–33. IEEE, 2008.
12. Suhit Gupta, Gail Kaiser, David Neistadt, and Peter Grimm. DOM-based content extraction of HTML documents. In *Proceedings of the 12th international conference on World Wide Web*, pages 207–214. ACM, 2003.
13. Jonathan Hedley. Jsoup HTML parser, 2009.
14. Rong Jin, Alex G Hauptmann, and ChengXiang Zhai. Language model for information retrieval. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 42–48. ACM, 2002.
15. Hung-Yu Kao, Jan-Ming Ho, and Ming-Syan Chen. Wisdom: Web intrapage informative structure mining based on document object model. *IEEE Transactions on Knowledge and Data Engineering*, 17(5):614–627, 2005.
16. Christian Kohlschütter. A densitometric analysis of web template content. In *Proceedings of the 18th international conference on World wide web*, pages 1165–1166. ACM, 2009.
17. Christian Kohlschütter et al. Boilerpipe – boilerplate removal and fulltext extraction from HTML pages. *Google Code*, 2010.

18. Christian Kohlschütter, Peter Fankhauser, and Wolfgang Nejdl. Boilerplate detection using shallow text features. In *Proceedings of the third ACM international conference on Web search and data mining*, pages 441–450. ACM, 2010.
19. Victor Lavrenko and W Bruce Croft. Relevance based language models. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 120–127. ACM, 2001.
20. Shian-Hua Lin and Jan-Ming Ho. Discovering informative content blocks from web documents. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 588–593. ACM, 2002.
21. Hadi Mohammadzadeh, Thomas Gottron, Franz Schweiggert, and Gholamreza Nakhaeizadeh. Extracting the main content of web documents based on character encoding and a naive smoothing method. In *Software and Data Technologies*, pages 217–236. Springer, 2011.
22. Melanie Neunerdt, Eva Reimer, Michael Reyer, and Rudolf Mathar. Enhanced web page cleaning for constructing social media text corpora. In *Information Science and Applications*, pages 665–672. Springer, 2015.
23. Miroslav Spousta, Michal Marek, and Pavel Pecina. Victor: the web-page cleaning tool. In *4th Web as Corpus Workshop (WAC4)-Can we beat Google*, pages 12–17, 2008.
24. Fei Sun, Dandan Song, and Lejian Liao. Dom based content extraction via text density. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, pages 245–254. ACM, 2011.
25. Karane Vieira, Altigran S Da Silva, Nick Pinto, Edleno S De Moura, Joao Cavalcanti, and Juliana Freire. A fast and robust method for web page template detection and removal. In *Proceedings of the 15th ACM international conference on Information and knowledge management*, pages 258–267. ACM, 2006.
26. Lan Yi, Bing Liu, and Xiaoli Li. Eliminating noisy information in web pages for data mining. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 296–305. ACM, 2003.