Boilerplate Detection using Shallow Text Features

Christian Kohlschütter, Peter Fankhauser, Wolfgang Nejdl

L3S Research Center / Leibniz Universität Hannover
Appelstr. 9a, 30167 Hannover
Germany
{kohlschuetter, fankhauser, nejdl}@L3S.de

ABSTRACT

In addition to the actual content Web pages consist of navigational elements, templates, and advertisements. This boilerplate text typically is not related to the main content, may deteriorate search precision and thus needs to be detected properly. In this paper, we analyze a small set of shallow text features for classifying the individual text elements in a Web page. We compare the approach to complex, state-of-the-art techniques and show that competitive accuracy can be achieved, at almost no cost. Moreover, we derive a simple and plausible stochastic model for describing the boilerplate creation process. With the help of our model, we also quantify the impact of boilerplate removal to retrieval performance and show significant improvements over the baseline. Finally, we extend the principled approach by straight-forward heuristics, achieving a remarkable accuracy.

Categories and Subject Descriptors

H.3.3 [**Information Systems**]: Information Search and Retrieval

General Terms

Algorithms, Experimentation, Theory

Keywords

Boilerplate Removal, Template Detection, Full-text Extraction, Web Document Modeling, Text Cleaning

1. INTRODUCTION

When examining a Web page, humans can easily distinguish the main content from navigational text, advertisements, related articles and other text portions. A number of approaches have been introduced to automatize this distinction, using a combination of heuristic segmentation and features. However, we are not aware of a systematic analysis of which features are the most salient for boilerplate content. In this paper, we analyse the most popular features used for boilerplate detection on two corpora. We show that a combination of just two features - *number of words*

and *link density* - leads to a simple classification model that achieves competitive accuracy. The features have a strong correspondence to stochastic text models introduced in the field of Quantitative Linguistics. Moreover, we show that removing boilerplate content based on these features significantly improves precision on the BLOGS06 benchmark, at almost no cost.

The paper is structured as follows. After shortly reviewing related work in Section 2 we discuss potential features for detecting boilerplate content in Section 3. Section 4 describes our content classification experiments, which we performed in two flavors: the two-class problem for boilerplate/content and a four-class problem specific for the news domain. In Section 5 we give a statistical linguistic interpretation of our observations. In Section 6 we apply the established model to the problem of Information Retrieval and show that precision can significantly be improved. Section 7 concludes with a discussion of further work.

2. RELATED WORK

Approaches to boilerplate detection typically exploit DOM-level features of segments by means of handcrafted rules or trained classifiers, or they identify common, i.e., frequently used segments or patterns/shingles on a website [3, 8, 9, 14, 24]. Using a combination of approaches, Gibson et al. quantify the amount of template content in the Web (40%-50%) [14]. Chakrabarti et al. determine the "templateness" of DOM nodes by a classifier based upon regularized isotonic regression [6] using various DOM-level based features, including shallow text features as well as site-level hyperlink information. Yi et al. simplify the DOM structure by deriving a so-called Site Style Tree which is then used for classification [26]. Baluja [2] employs decision tree learning and entropy reduction for template detection at DOM level.

Template detection is strongly related to the more generic problem of web page segmentation, which has been addressed at DOM-level [7], by exploiting term entropies [17] or by using Vision-based features [5]. Kohlschütter et al. present a statistical model for the distribution of segment-level text densities, and use the text density ratios of subsequent blocks to identify page-level segments [18, 19]. The CleanEval competition [4] aims at establishing a representative corpus with a gold standard in order to provide a transparent and comparable platform for boilerplate removal experiments. The evaluated algorithms mainly apply machine learning techniques for the classification [4]. For instance, NCleaner [10] utilizes a trained n-gram based language model, and Victor [23] employs a multi-feature sequence-labeling approach

based on Conditional Random Fields, similar to the approach of Gibson et al. [15]. Another CleanEval contestant, BTE, determines the largest contiguous text area with the least amount of HTML tags and marks it as "full text" [12, 13]. The heuristic is based on the observations that the *tag density* within boilerplate text is higher than within fulltext content and that main content usually is longer than boilerplate text. A similar approach, which uses an n-gram model plus several HTML-based heuristics, mainly focusing on extracting the main content of news articles, has recently been presented by Pasternack et al. [21] and also evaluated against the CleanEval benchmark, apparently with high accuracy. We analyse a representative set of features used by these approaches for automatic boilerplate classification.

One driving motivation for boilerplate text detection is to improve web search and mining, similar in spirit to simple stop-word removal. Viera et al. [24] introduce an approach based on detecting common subtrees in a few sample pages similar to [26] and observe that clustering and classification accuracy can be improved significantly by removing such common subtrees. Fernandes et al. [11] measure the importance of blocks by a combination of average inverse site frequency of terms in a block, as a measure for block commonality, and the similarity of a block with other blocks on the same page. By weighting terms by their block importance they significantly improve accuracy over the baseline BM25. We show that densitometric features, which can be computed efficiently online, without resorting to global frequencies, also significantly improves retrieval accuracy.

3. WEB PAGE FEATURES

3.1 Feature Levels

Many features that can be used for the classification of Web page segments have already been described [15, 16, 23, 26]. It is generally expected that the combination of several features can be used to identify text fragments as *headline*, *full text*, *enumeration*, *navigation*, *disclaimer notice* etc., which can then be separated into content and boilerplate text. The number of potential dimensions for this task is huge: text-based strategies like n-gram models can result in tens of thousands of relevant features, which apparently makes the classifier susceptible to overfitting to content and layout of a particular subset.

In search of a domain independent, Web-scale solution, we avoid these token-level features altogether.

Features may be extracted at four different levels: Individual text blocks (elements), the complete HTML document (a sequence of one or more text blocks plus structural information), the rendered document image (the visual representation as in a Web browser) and the complete Web site (i.e., the collection of documents which share a common layout and wording). While the former two levels can apparently be examined for each document locally, the latter two require external information, such as images and CSS definitions for the rendering process and, in order to statistically determine site-level templates and boilerplate text, a sufficiently large number of pages from the same website.

Using features from the two external levels may be highly beneficial to the classification accuracy iff the corresponding data is available. However, there are two major drawbacks. First, rendering pages for classification is a computational expensive operation. Second, template statistics need to be learned separately for each web site, they usually cannot be re-used for another website layout. Moreover, it is questionable whether such models are then domain independent (or trained for the news domain only, for instance). We therefore avoid these levels except for one reference feature: the frequency of the text in the whole corpus. Using this feature we can identify phrases commonly used in boilerplate.

3.2 Structural Features

Many approaches for Web page segmentation and intra-document text classification utilize structural features in Web pages, such as individual HTML tags (headline, paragraph, anchor text link, image, etc.) or sequences/nested subtrees of HTML tags as well as the presence of particular CSS classes and styles. Of note, the more CSS is used, the less important the semantics of an HTML tag becomes – it is perfectly legal to only use DIV tags and describe the "semantics" of a particular division using style-sheet classes. Unfortunately, CSS classes and sequences of HTML tags are inherently site- and document-specific. Moreover, to fully interpret these rules one has to essentially render the page.

As we want to avoid site-specific signals (which may lead to over-fitting to a particular data set or domain) as well as a costly rendering of pages, we only examine the following structural features: The presence of a particular headline tag (`H1`, `H2`, `H3`, `H4`, `H5`, `H6`), a paragraph tag (`P`), a division tag (`DIV`) and the anchor text tag (`A`) as an HTML element that encloses a particular text block.

3.3 Shallow Text Features

Because boilerplate detection does not inspect text at the *topical* level but rather at the *functional* level, we do not consider the bag of words as classification features. An evaluation at token-level may provide skewed results that describe a particular domain only. Instead, we examine shallow text features at a higher, domain- and language-independent level, which have been discussed in the field of Quantitative Linguistics: *Average word length* (in our definition words are white-space delimited character sequences which at least contain one letter or digit), *average sentence length* (the sentence boundaries are identified by a simple pattern-based heuristic checking for the presence of full stops, question or exclamation marks as well as semi-colons) and the absolute number of words.

Another important source for the classification task is the local context, i.e., the absolute and relative position of a text block in the document. If the segmentation granularity is high, it is likely that full-text is followed by full-text and template is followed by template. Moreover, when there is a significant amount of boilerplate text, the main content usually is surrounded by boilerplate (header, footer, left-navigation, right-navigation etc.), not vice versa (i.e., even if the very last text block contains a sentence, if it is a copyright or disclaimer notice, it is regarded boilerplate).

We also examine a few heuristic features: the absolute number of words that either start with an uppercase letter or are completely upper-case as well as the ratio of these words compared to the total number of words and the ratio of full stops to the overall number of words, the number of date/time-related tokens and the number of vertical bars "|" (these characters can sometimes be found in navigational boilerplate text). Moreover, we also compute the link density (called *anchor percentage* in [15]), as the number of to-

kens within an `A` tag divided by the total number of tokens in the block; for this computation we do not regard the `A` tag as a block separator.

3.4 Densitometric Features

Besides the link density measure, we also evaluate the text density of each particular block. The text density measure $\varrho(b)$ has been introduced in [19] and utilized for segmenting web pages in a merge-only strategy called *Block Fusion*. There, adjacent text fragments of similar text density (interpreted as "similar class") are iteratively fused until the blocks' densities (and therefore the text classes) are distinctive enough. Using various settings, including a rule-based approach, it was shown that the resulting block structure closely resembles a manual segmentation.

Text density was derived from the pixel-based text density of Computer Vision-based approaches and transformed to character level. Basically, it counts the number of tokens $|T(b)|$ in a particular text block b divided by the number of lines $|L(b)|$ covered after word-wrapping the text at a fixed column width w_{max} (the empirically estimated optimal value for English text is between 80 and 90 characters; in this paper, we use $w_{max} = 80$). Due to the side-effect of having an incompletely filled last line after wrapping, the latter is not taken into consideration unless it is the only line in the segment:

$$T'(b) = \{t \,|\, t \in T(l), \ l_{first}(b) \le l < l_{last}(b)\}$$

$$\varrho(b) \ = \ \begin{cases} \frac{|T'(b)|}{|L(b)|-1} & |L(b)| > 1 \\ |\,T(b)\,| & \text{otherwise} \end{cases} \qquad (1)$$

4. CLASSIFICATION EXPERIMENTS

4.1 Goals and Approach

The goal of this section is to analyse the features introduced in Section 3 for boilerplate detection.

The overall approach is simple: Web pages are segmented into atomic text blocks, which are then annotated with features and on this basis classified into content or boilerplate. Atomic text blocks are sequences of character data which are separated by one or more HTML tags, except for `A` tags – in order to compute the link density.

To train and test classifiers for various feature combinations we start from a known text domain: news articles on the Web. The domain is large and diverse because numerous independent sources contribute to it, is readily available for analysis (e.g. from a news search engine) and the structure is well-understood: Usually one news article (consisting of one or more headlines, the article body and supplemental information like fact boxes, image captions etc.) is surrounded by the standard layout of the publisher's web site (linked headlines and teasers to other news articles, related or not, advertisements, copyright notices etc.). In some cases, the publishers also allow users to comment on the article, comments may then appear on the page nearby the article.

We have labeled a representative subset of news articles from different sites with different layouts according to the observed text types (boilerplate/content as well as other classes like headline, user comments etc.) The labeled set is then split into a training and a test set (using a 10-fold cross validation) and fed into a classifier (we use decision trees and

Class	# Blocks	# Words	# Tokens
Total	72662	520483	644021
Boilerplate	79%	35%	46%
Any Content	21%	65%	54%
Headline	1%	1%	1%
Article Full-text	12%	51%	42%
Supplemental	3%	3%	2%
User Comments	1%	1%	1%
Related Content	4%	9%	8%

Table 1: Class-Distribution in the GoogleNews set

linear support vector machines) to measure the accuracy of the approach. To analyze domain independence, we also evaluate the classifiers against datasets from other domains.

4.2 Datasets and Gold Standard

Our evaluation is performed on two datasets, a news collection for training and testing and a cross-domain collection for validation.

News Collection. The news collection consists of 621 manually assessed news articles from 408 different web sites. The news articles were sampled randomly from a larger crawl of 254,000 articles from 7,854 web sites which we acquired by monitoring the Google News search engine during the first half of 2008. We monitored the news headlines of six different English-speaking Google News portals (USA, Canada, UK, South Africa, India, Australia) and four categories (World, Technology, Sports, Entertainment) and fetched the full text HTML of the corresponding linked articles. The ranked distribution of articles per web site apparently is power-law distributed (maximum number of articles per host: 3774, averge: 32.38, median: 5). The top-5 hosts are *ap.google.com*, *afp.google.com*, *reuters.com*, *iht.com*, *news. bbc.co.uk*; at the break-even between rank and frequency (200) is *en.rian.ru* whereas sites like *financeasia.com* and *photonicsonline.com* appear at the bottom. In the examined subset, the maximum number of articles per host is 12 (*news.com.au*) whereas the average and median are 1.52 and 1 respectively. In this paper, we use the term "GoogleNews" to describe that subset. In the following sections we focus on this collection except for the frequency of text blocks, which we compute from the complete crawl.

Using a Web browser based text annotator, for each HTML page in the GoogleNews set seven human assessors labeled[1] sequences of text as either *headline*, *fulltext*, *supplemental* (text which belongs to the article but is not fulltext, such as image captions etc.), *user comments*, *related content* (links to other articles etc.). Unselected text is regarded *not content* (boilerplate). The labels were then stored at the level of individual text blocks (i.e., any character sequence that is not interrupted by an HTML tag, except the `A` tag, as described in Section 3.3). The distribution of classes at three different levels is depicted in Table 1; we counted the number of text blocks, words and unfiltered tokens (including non-words) separately. The raw data and the gold standard are available online.[2]

Cross-Domain Collection. The CleanEval collection is a benchmark corpus created particularly for the eponymous boilerplate removal competition of the ACL Web-as-Corpus

---
[1]Every page was assessed only once, as we do not expect significant inter-assessor disagreement in this context.
[2]http://www.L3S.de/~kohlschuetter/boilerplate

community [4]. The collection consists of 798 raw HTML pages randomly sampled from Web search engines, from which 733 pages have already been manually assessed and split into a training set of 58 documents and a test set of 675 documents. The assessment was performed as follows. After converting the HTML document into plain text, all boiler-plate text has manually been removed and remaining text has been structurally labeled as *paragraph*, *headline* or *list element*. The raw data and the gold standard are available online.[3] Unfortunately, because the assessors worked with plain text that has been derived from a Browser-rendered representation of the documents, the gold standard cannot directly be used for an analysis at HTML level. We will nevertheless evaluate our approach against this benchmark to allow a comparison to other CleanEval contestants.

4.3 Evaluation

4.3.1 Training and Testing on GoogleNews

As we see from Table 1, the class distribution is strongly dominated by *Boilerplate* and *Article Full Text*; the other four classes quantitatively play a minor role. The class *User Comments* was only assessed to quantify the amount of comments text compared to the remaining full text; for the purpose of boilerplate detection we treat comments as main content. Because of the strongly skewed distribution of the initial six text classes, we evaluate a two-class problem (boilerplate vs. content) and a four-class problem (boilerplate vs. full-text/comments, headline, supplemental) separately. The four-class problem generally is more difficult to solve, so we expect lower accuracies here.

Using Weka, we examine the per-feature information gain and evaluate machine-learning classifiers based on Decision Trees (1R and C4.8) as well as Support Vector Machines (SMO). We measure classification accuracy by Precision, Recall, F_1-Score, False Positive Rate and ROC Area under Curve (AuC); all scores are normalized based on the number of words in a block, i.e., large blocks are weighted higher than small blocks. Figure 1 shows the features in decreasing order of their information gain. The information gain (Kullback-Leibler-divergence) is defined as the change in information entropy from the prior state (C) to a state that takes some information as given $(C|A)$: $IG(C, A) = H(C) - H(C | A)$ and helps identifying the most significant features. Generally, very simple features like *Relative Position*, *Average Word Length* and *Number of Words* of the current block appear to be strong indicators for class membership. For the sake of clarity, we will test the classification accuracy of these highly ranked features separately. Interestingly, the text-flow capturing variants of these features (*Number of Words Quotient*, *Text Density Quotient*, *Relative Position*), which relate the value of the current block to the previous one, provide the highest information gain, indicating that the intra-document context plays an important role; this is also substantiated by the classification results. Table 2 presents the evaluated algorithms and the achieved accuracies along with the number of features (dimensions) used and the number of leaves for all decision-tree-based algorithms.

The classification baseline is the *ZeroR* classifier, which in our case always predicts *Article Full-text* (4-class problem)

and *Content* (2-class problem). Due to the class weights this results in an ROC area-under-curve of less than 50%.

The *1R* classifier determines the feature with the least error rate and partitions the corresponding numeric values to derive simple classification rules (the number of partitions equal the number of leaf nodes in a decision tree). 1R over all features resulted in a simple rule with an acceptable accuracy: Any block with a text density less than 10.5 is regarded boilerplate. We analyzed the 1R partitioning also for the features *Average Sentence Length*, *Average Word Length*, *Link Density* and *Number of Words* and got similar (slightly lower) accuracies. However, *Average Word Length* is fairly unsuitable for classification, as 1R generates many partitions between average word length 4 and 5 which alternate between *Boilerplate* and *Article Content*.

We get promising results from the C4.8-based decision-trees, however at the cost of model complexity; in order to avoid overfitting, the algorithm has been configured to only consider leaves matching at least 1000 instances. By using all the 67 available features (including features from the previous and next blocks) we get a remarkable ROC AuC of 98% for the 2-class problem and 96.9% for the 4-class problem; we also achieved similar results using an SMO support-vector machine with a linear kernel.

By applying reduced-error pruning we were able to simplify the decision tree to only use 6 dimensions (2 features each for current, previous and next block) without a significant loss in accuracy (ROC AuC 96.9% for the 2-class problem), see Algorithms 1 and 2.

Algorithm 1 Densitometric Classifier

```
curr_linkDensity <= 0.333333
| prev_linkDensity <= 0.555556
| | curr_textDensity <= 9
| | | next_textDensity <= 10
| | | | prev_textDensity <= 4:  BOILERPLATE
| | | | prev_textDensity > 4:  CONTENT
| | | next_textDensity > 10:  CONTENT
| | curr_textDensity > 9
| | | next_textDensity = 0:  BOILERPLATE
| | | next_textDensity > 0:  CONTENT
| prev_linkDensity > 0.555556
| | next_textDensity <= 11:  BOILERPLATE
| | next_textDensity > 11:  CONTENT
curr_linkDensity > 0.333333:  BOILERPLATE
```

**Algorithm 2** Classifier based on Number of Words

```
curr_linkDensity <= 0.333333
| prev_linkDensity <= 0.555556
| | curr_numWords <= 16
| | | next_numWords <= 15
| | | | prev_numWords <= 4:  BOILERPLATE
| | | | prev_numWords > 4:  CONTENT
| | | next_numWords > 15:  CONTENT
| | curr_numWords > 16:  CONTENT
| prev_linkDensity > 0.555556
| | curr_numWords <= 40
| | | next_numWords <= 17:  BOILERPLATE
| | | next_numWords > 17:  CONTENT
| | curr_numWords > 40:  CONTENT
curr_linkDensity > 0.333333:  BOILERPLATE
```

---

[3]http://nlp.fi.muni.cz/~xpomikal/cleaneval/
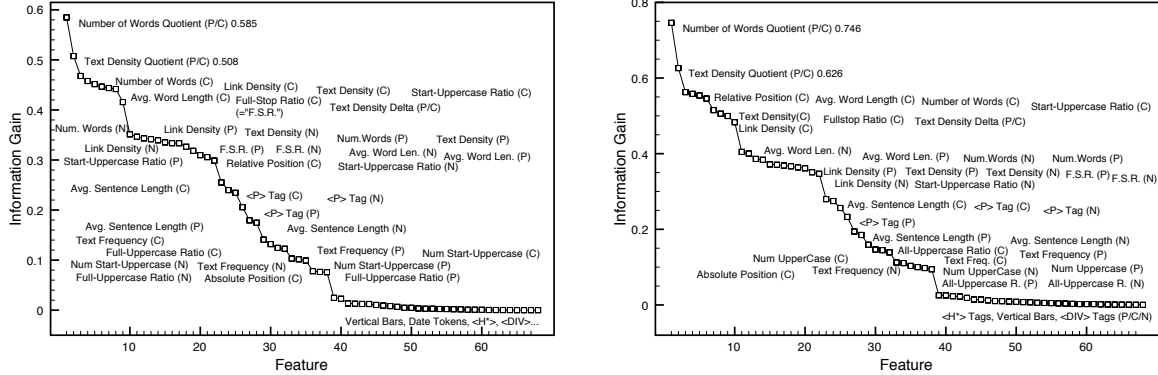
(a) 2-class problem (b) 4-class problem
Figure 1: Per-Feature Information Gain for the GoogleNews collection.

Algorithm	Dim	Precision		Recall		F_1-Score		FP Rate		ROC AuC		Leaves	
ZeroR (baseline; predict "Content")	0	40.7	35.0	63.8	59.2	49.7	44.0	63.8	59.2	49.0	48.9	–	–
Only Avg. Sentence Length	1	78.5	72.4	67.9	66.6	68.0	65.2	21.4	22.3	73.3	72.1	2	4
C4.8 Element Frequency (P/C/N)	3	77.7	70.9	76.2	73.2	73.8	69.8	38.3	34.7	70.9	69.8	9	4
Only Avg. Word Length	1	80.2	77.4	77.0	74.8	77.5	73.5	19.5	20.0	78.8	77.4	2	178
Only Number of Words @15	1	86.7	80.9	86.7	84.9	86.7	82.8	15.5	15.5	85.6	84.7	2	2
Only Link Density @0.33	1	88.5	81.7	87.8	83.8	87.4	81.4	19.2	22.5	84.3	80.7	16	2
1R: Text Density @10.5	1	87.8	81.4	87.9	85.4	87.9	83.4	14.3	15.3	86.8	85.0	2	2
C4.8 Link Density (P/C/N)	3	91.1	83.7	91.1	87.4	91.0	85.4	12.1	14.3	94.2	90.8	37	32
C4.8 Number of Words (P/C/N)	3	91.1	87.6	90.8	89.3	90.9	87.6	8.9	1.0	94.7	94.6	40	48
C4.8 All Local Features (C)	23	92.9	88.7	92.9	89.9	92.9	88.7	8.7	10.6	96.6	95.7	102	72
C4.8 NumWords + LinkDensity, simplified	**6**	**92.2**	**84.8**	**92.2**	**88.9**	**92.2**	**86.8**	**10.1**	**12.1**	**95.7**	**93.3**	**8**	**7**
C4.8 Text + LinkDensity, simplified	**6**	**92.4**	**84.7**	**92.4**	**88.8**	**92.4**	**86.7**	**8.5**	**11.4**	**96.9**	**93.2**	**8**	**12**
C4.8 All Local Features (C) + TDQ	25	92.9	89.2	93.0	90.3	92.9	89.1	8.3	9.4	97.2	96.1	109	78
C4.8 Text+Link Density (P/C/N)	**6**	**93.9**	**89.3**	**93.8**	**91.0**	**93.9**	**89.5**	**6.7**	**8.4**	**97.6**	**96.1**	**51**	**45**
C4.8 All Local Features (P/C/N)	64	95.0	91.3	95.0	92.4	95.0	91.4	5.5	7.1	98.1	96.7	105	98
C4.8 All Local Features + Global Freq.	**67**	**95.1**	**91.5**	**95.0**	**92.5**	**95.1**	**91.6**	**5.4**	**6.8**	**98.0**	**96.9**	**99**	**125**
SMO All Local Features + Global Freq.	67	95.3	92.4	95.3	93.2	95.3	91.9	5.4	6.8	95.0	94.0	-	-

Table 2: Weka Classification Accuracies for the Google News Collection (2-class/4-class problem)

### 4.3.2 Application to CleanEval and Re-Validation

To test the domain-independence of the determined classifiers, we applied the two simplified C4.8 classifiers to the CleanEval collection. We evaluated the 2-class problem (boilerplate vs. content of any kind, called *TO* in CleanEval) for the classifier that has been trained for the GoogleNews collection and one that has been trained on the CleanEval training set. In the latter case, the decision rule was even simpler: accept all blocks that have a minimum text density of 7 and a maximum link density of 0.35.

Because the CleanEval collection only provides assessments and the algorithmic output at text-level, we cannot directly reuse the setup we used for the GoogleNews evaluation. The CleanEval initiative provides their own accuracy measure which is based upon a weighted Levenshtein Edit Distance at token-level [4]. The computation is expensive and also not essential for this task. We confirmed that the scores can be approximated well with the much simpler bag-of-words token-level F_1 score (like in the GoogleNews setup, except that class weights are not taken into account). As our scores therefore slightly differ from the ones in [4], we re-evaluated the available results of three CleanEval contestants (BTE, Victor, NCleaner) and also added the heuristics

by Pasternack et al. [21] (in two flavors, the unigram model trained on CleanEval and the trigram model trained on a news corpus) to the set of competitors, as well as a baseline ("Keep all text") and a classifier solely based on the feature with the highest information gain: number of words; we mark every block with at least 10 words as content.

The average (μ) and median (m) as well as the ranked accuracy for each evaluated strategy are depicted in Figure 2a. We see that the two flavors of the Pasternack heuristic drastically differ in terms of accuracy. We assume that the algorithm needs proper training to succeed for a particular corpus, and the trigram model from the news domain was not generic enough for the CleanEval dataset.

Additionally, to understand how far heuristic additions could further improve the classification, we extended our two decision tree classifiers downstream with hand-crafted rules. In one extension, we only take the content block with the highest number of words (*Largest Content Filter*). In another extension, we add rules that are specific for news (*Main Content Filter*). It extends Largest Content Filter by removing any text that is below a clearly identifiable comments section (a block solely containing one out of 11 indicator strings like "User comments:" etc.) and above a clearly
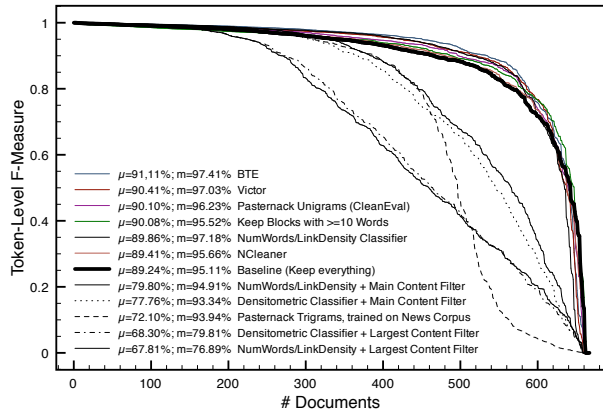
(a) CleanEval
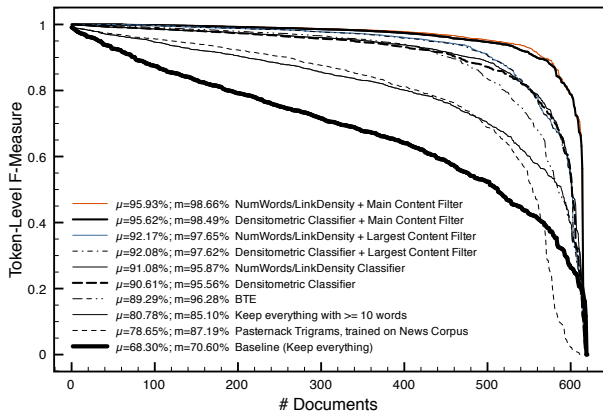
(b) GoogleNews

Figure 2: Performance of Boilerplate Detection Strategies

Figure 3: CleanEval Text Density Distribution

identifiable title (derived from the HTML document's TITLE value). As we can see from Figure 2a, for the CleanEval collection these modifications resulted in much worse results than the baseline. On the other hand, the very simple strategy to keep all blocks with at least 10 words (as well as our NumWords/LinkDensity classifier) performed just as good as the Pasternack unigram and the NCleaner setups that have specifically been trained for CleanEval.

Ultimately, we see that basically keeping all text (i.e. not removing anything) would be a good strategy, which only is marginally worse than the apparently best solution (BTE)! This leads to the question whether there were failures in the assessment process, whether the collection is comparable to the GoogleNews collection or at all appropriate for the purpose of boilerplate detection.

We repeated this evaluation for the GoogleNews collection, computing accuracy scores in the same way using the same algorithms (BTE as the alleged winner for CleanEval, Pasternack Trigrams as a supposedly mediocre strategy and the algorithms we introduced in this paper). For the Pasternack algorithm, we used the Web service provided by the authors; unfortunately, there was no unigram implementation available. Figure 2b shows the corresponding results.

We see that the baseline for GoogleNews is much lower than for CleanEval; all tested algorithms perform differently and are usually better than the baseline, except for the Pasternack strategy, which underperformed in a few cases.
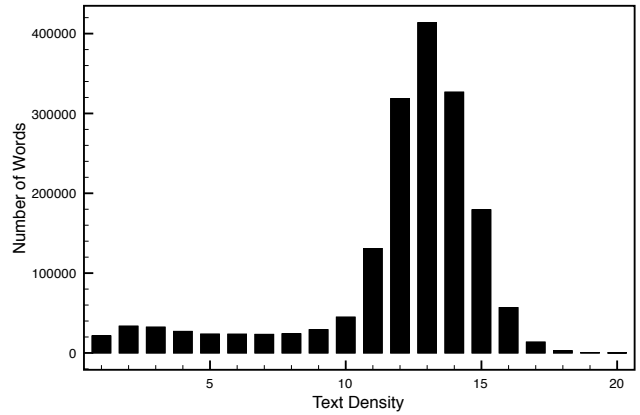
Overall its performance is lower than expected, given the fact that it has been trained on news articles. We can only assume a bug in their implementation or high overfitting towards a particular subset of news sites. The strategy to just keep everything with a minimum of 10 words did not work very well either, although better than the Pasternack trigrams and, on average, improves the baseline by 18.3%. BTE is on par with our two simplified classifiers (using *text density* and *number of words* respectively); it is a little bit better for the median but worse on average. Our classifier based on the number of words per block and its link density yields improve the baseline by 33.3%.

In the end, the use of the two heuristic filters (*Largest/-Main Content Filter*) can further improve the detection accuracy for the GoogleNews dataset to an almost perfect average F_1 score of 95.93% (this is a 40% relative improvement over the baseline). Even though we see that these algorithms failed for CleanEval, we expect them to work generically for the news domain. On the other hand, we see that both, BTE and our two simplified classifiers work quite well for both collections. Of note, our classifier is far more efficient than BTE. It runs in linear time, whereas BTE has a quadratic upper bound. Furthermore, it can return more than a single piece of text. BTE's assumption that only one block (the largest having the least tag density) completely covers the main content seems not to hold for all cases: compared to the densitometric classifier, BTE only achieves a suboptimal accuracy in our news experiment. This discrepancy may be due to the fact that the CleanEval collection actually contains less boilerplate text than all other collections we examined (only very little words are in blocks with a text density lower than 11, see Figure 3).

5. QUANTITATIVE LINGUISTIC ANALYSIS

5.1 Text Density vs. Number of Words

In all cases but one, the classifier using *Number of Words per Block* performed slightly better than the classifier using *Text Density*. Also it seems sufficient for a good classification. To get a better understanding why this strategy performs so well, we need to analyze the created decision tree rules (see Algorithms 1 and 2). We see that the two classifiers do not differ for the link density-specific rules; if the text block consists of more than 33% linked words, it is most likely boilerplate, unless the block is surrounded by long/dense text blocks.
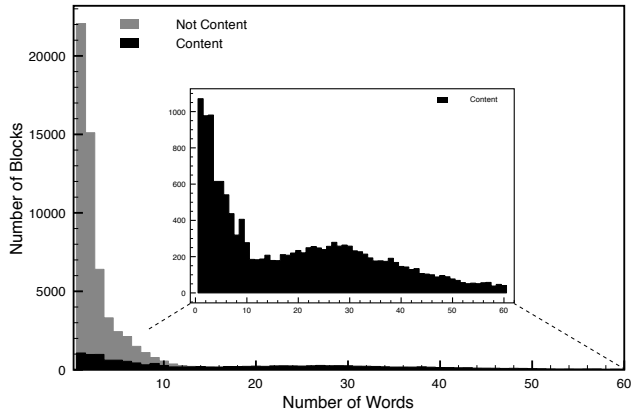
Figure 4: Number of Words Distribution (GoogleNews)

Figure 5: Text Density Distribution by class (GoogleNews)

Actually it is likely that both measures, *text density* as well as *number of words* describe the same fundamental principle of text, which however is more visible through text density than through the plain number of words. As the absolute number of words theoretically is unbounded (the longest block in the GoogleNews collection consisted of 1122 words), yet dominated by boilerplate (79% of all blocks, see table 1), a straight visual distinction between boilerplate and content is hard to spot (Figure 4; we magnified the "content" part for clarity). Also, the rules generated for the number-of-words classifier are difficult to interpret. It is unclear, for instance, why everything exactly above *40* words is regarded full text and not already at *30* etc. However, if we look at the very same data using the text density metric (rounding the density to the nearest integer $[\varrho(b)]$), we can clearly identify three modes of a mixed distribution (see Figure 5).

In [18], Kohlschütter showed for a representative Web corpus (Webspam-UK 2007, ham-part) that the distribution of words in blocks with a particular text density can effectively be modeled as a combination of two beta distributions and a normal distribution. Each beta distribution was assumed to represent one class of text, "full-text" content (complete sentences) and "template text" (incomplete sentences); the normal distribution acts as a fuzzy transition between them. In fact, we were able to apply the same model to the GoogleNews collection data, with a very high goodness of fit ($R^2 = 0.997$, RMSE = 0.0213). As opposed to Kohlschütter's initial results on an unlabeled corpus, we now have manually labeled annotations, so we can finally examine his hypothesis at a higher level of confidence.

Indeed, the main article's full text, as well as the user comments and, to some degree, supplemental text, can basically be described as blocks with a text density ≥ 10 (in fact, the 1R algorithm suggested this split, achieving a ROC AuC of 86.8%). The remaining blocks with a lower density almost completely describe boilerplate text (headlines appear at text density 4 and higher; some "supplemental text" may expose an even lower text density). Moreover, the fuzzy class seems to be strongly dominated by linked text (hypertext), which might explain why the addition of the link density feature significantly improves the classification accuracy.

Obviously, the text density measure helps us to visualize the mixed distribution in a compact way (much better than
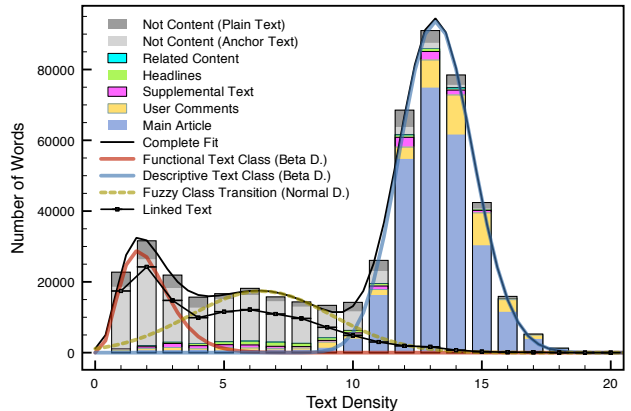
the absolute number of words, see Figure 4), even though it appears that for the actual purpose to separate and to classify the two types of text (template and fulltext) the *number of words per block* are sufficient.

Actually we can approximate the density distribution for visualization purposes solely using the number of words as follows. From the definition of text density (Equation 1) we see that two cases are differentiated: wrapped text (i.e. covering more than one line) and unwrapped text (i.e. only one line). If the line is wide enough (we used 80 characters), all densities below a certain number of words λ describe one-line blocks (except for the unusual case where blocks contain very long words), or combinations thereof. In order to reach a line wrap boundary, a certain number of words need to be written, and thus a large text density score indicates a larger number of words. In fact, the difference of the number of words contained in blocks with at least $\lambda = 11$ words (345.175) to the number of words contained in blocks with a text density of at least 11 (358.428) is insignificant (3.8%).

5.2 Stochastical Text Model

The fairly clear separation between short boilerplate and longer content blocks with respect to text density suggests a simple generative process:

First, let us find a sufficiently good model for the overall process of generating words. We can see the creation process of text blocks as a Shannon random writer [22].

Imagine the author decides with some probability to write a word or to finish the current block and proceed to the next one. This essentially is a first-order Markov process with two states, T (*add another word to the text*) and N (*skip to the next block*); see Figure 6a. The probability of staying in the same state is always the complementary probability of moving to the other state. As we can regard subsequent newlines as a single operation, we have $P_N(N) = 0$ and thus $P_N(T) = 1$ (after skipping to a new block, always at least one word is written).

The state transitions from T can be modeled as a simple Bernoulli trial. Consider the transition to N as *success* (p) and the emission of another word as failure $(1 - p)$. The probability that there are k failures (for $k = 0, 1, 2, 3, ...$) before the first success is $Pr(Y = k) = (1 - p)^k p$.

Coming from state N means we already have emitted one word, so the actual probability for emitting x words ($k - 1$ failures) in the simple scenario then is

$$Pr(Y = x) = (1 - p)^{x-1} \cdot p = P_T(T)^{x-1} \cdot P_T(N) \quad (2)$$

which is the 1-displaced geometric distribution; it has extensively been discussed in the field of Quantitative Linguistics [1]. While there are more sophisticated, better matching models for describing this process, the geometric distribution is a good starting point, particularly at corpus level where the individual variations of certain authors become indistinct. Applied to the GoogleNews corpus, for $P_T(N) = 0.3145$ we achieve a goodness of fit of $R_{adj}^2 = 96.7\%$ with a root mean square error ($RMSE$) of 0.0046.

Let us now add the two different types of text that we have discovered in our evaluation, *short* and *long* text. We extend the simple random writer by transforming the state T into two separate states S (print a word of short text) and L (print a word of long text), see Figure 6b. As L and S replace T, the probabilities to arrive at L or S coming from N must sum up to $P_N(T) = 1$. Once in the state L or S, we again have a Bernoulli trial: either continue producing words (all of long or short text respectively, no intra-block mixtures) or terminate and go back to state N. As we expect from short text to terminate quickly after a few words and from long text to terminate after a higher number of words, we require that $P_S(N) \gg P_L(N)$.

In this mixed scenario, the probability density distribution therefore is:

$$Pr(Y = x) = P_N(S) \cdot \left[P_S(S)^{x-1} \cdot P_S(N)\right] +$$
$$+ P_N(L) \cdot \left[P_L(L)^{x-1} \cdot P_L(N)\right] \quad (3)$$

Applying this model to the GoogleNews data results in a higher goodness of fit of $R_{adj}^2 = 98.81\%$ with $RMSE = 0.0027$ for $P_N(S) = 1 - P_N(L) = 0.7586$, $P_S(N) = 0.3968$ and $P_L(N) = 0.04369$, which supports our assumption of the mixture, even if the geometric distribution only is a rough approximation. As the geometric distribution's expected value is defined as $E(x) = p^{-1}$ (short text has its mean at $1/0.3968 = 2.52$, long text at $1/0.04369 = 22.89$) and the determined probability $P_N(S) = 76\%$ is close to the assessed 79% (amount of blocks classified as boilerplate, see Table 1), we may attribute a large extent of short text to the *Boilerplate* class and most long text to the *Content* class (see Figure 4).

5.3 Linguistic Interpretation

The observed compound distribution can be regarded not of arbitrary nature but of a stochastical, quantitative linguistic one, implying that actually two different classes (strata) of text are embedded in the Web content. In the field of Quantitative Linguistics it is generally assumed that text creation process can be modeled as *urn trials* at the level of various linguistic units such as *phoneme*, *word*, *sentence*, *text segment* etc. and for several shallow features such as *frequency*, *length, repeat rate, polysemy* and *polytextuality* [25]. Even though it is still unclear by which exact parameters this process is driven, we can model it as a Bernoulli process. Through empirical experiments and simple stochastical concepts, we have shown that this model can be applied to describe the process of content creation on a Web page.
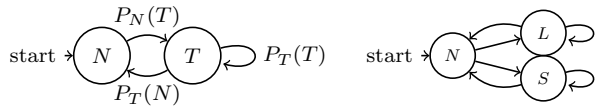
(a) Simple   (b) Mixed

Figure 6: Random Writer Models

While we cannot explain the reasons for chosing short or long text at some particular point in a document, we can interpret the statistically observed behaviour at corpus level. When composing a Web page, an author chooses with some probability whether she wants to write a sequence of actual full sentences or navigational elements. The choice surely depends on the context (hence we observe an improvement when the features from the previous and next block are taken into account, and this is why the geometric distribution does not fit perfectly). The use of full sentences usually means the author wants to make a more or less complex statement which needs grammatical constructs, long explanations etc. Extensive coding is required because the author (sender) does not expect that the audience (receivers) understand the information without explanation. This kind of text (*long text*) therefore can be regarded of *descriptive* nature (i.e., supplying the reader with the subject matter's details at the cost of higher syntactic complexity, just like the full text of this paper). The second kind of text (*short text*), grammatically incomplete or simple sentences, consisting of only a few words, is used whenever a quick, economic coding is possible, i.e. when the audience is expected to perceive und understand the encoded information without large effort (e.g., "Contact us", "Read more"). Such text is often used for headlines and navigational text (one kind of boilerplate). We can therefore regard the latter form of text of *functional* nature. While there are noticeable exceptions, it appears that, at least for the Web, there is a strong correlation between *short text* and *boilerplate text* as well as between *long text* and *content text*, which explains why the simple classification works so well.

These two strata of text can be visualized in a compact form through the text density measure because it is mostly irrelevant how many words an author spends within an individual text. As soon as she writes complete, non-trivial sentences (i.e., more than ca. 10 words in English) the produced text most likely falls into the *descriptive* class. Text density exactly provides this value-limiting boundary. By word-wrapping text at a predetermined line width (which is dependent upon the average sentence length in characters) and dividing the number of words by the number of lines, we literally "construct" this two-fold distribution and thus can better visualize what was already present in the raw number of words. An incomplete sentence will never wrap to more than one line (in this case text density equals to the number of words), whereas text consisting of complete sentences will always wrap, be averaged to the "typical" number of words in a sentence and encoded as a density value of a rather limited range. This limited range can then be better visualized histographically, as demonstrated.

To our best knowledge, the distinction between short and long text has not been discussed in the quantitative linguistics context so far. This is probably because the amount of

short text in the previously analyzed "offline works" is almost negligible (this also holds for the present paper). On the other hand, we expect to find higher amounts of short text in brochures, tabloids etc. An in-depth analysis of such content needs to be conducted as future work.

6. RETRIEVAL EXPERIMENTS

6.1 Setup

In this section we quantify the impact of boilerplate detection to search. The obvious assumption here is that boilerplate not only is another sort of text, it may also deteriorate search precision, particularly in those cases where keywords match "related articles" text or other keywords that are non-relevant to the actual main content. To evaluate our hypothesis for a representative scenario, we examine yet another domain of Web documents: Blogs.

Blogs are particularly relevant for this task because we may expect many links from one blog page to other blog entries, being topically or temporally related, and those links often include headlines and teaser texts of the referenced item. In addition to that, a TREC reference collection already exists, containing 3 million permalink documents retrieved from 100.000 different feeds, along with test queries (consisting of one to five words) and document assessments at (TREC'06 Blog Track, [20]) which were mainly used for measuring opinion retrieval performance. They used graded relevance scores (not relevant, topically relevant and three levels indicating positive, negative and mixed opinions). We will use this collection for our evaluation. We indexed the BLOGS06 collection using the Lucene IR library. Separate parallel indexes were created for document blocks with a particular number of words or a particular text density; this allows a selection of permitted ranges at runtime without reindexing. If our assumption holds, one can expect an improvement of the search precision when only words of the *descriptive text* class (long text) are considered for search.

6.2 Evaluation

We perform 50 top-k searches (with $k = 10$) for the queries defined in the TREC'06 Blog Track and evaluate precision at rank 10 (P@10; true/false relevance as in the TREC'06 benchmark results) as well as the normalized discounted cumulative gain ($NDCG_{10}$, graded relevance scores as present in the TREC'06 assessments). Using the standard Lucene ranking formula we perform 50 searches from queries predefined in the TREC'06 Blog Track and count the number of documents in the top-10 results which have been marked relevant by the TREC assessors. We repeatedly issue the queries for a sliding minimum text density between 1 and 20 and a sliding minimum number of words from 1 to 100 respectively (a minimum of 1 word equals to the baseline). As we only remove *short text*, there is no need for a *maximum* bound. We benchmark the performance of the BTE algorithm for this task and compare P@10 as well as $NDCG_{10}$ to our solution. Finally we compare the results to the P@10 scores reported from TREC'06.

Using the sliding minimum *Text Density* we are able to significantly improve precision (the baseline results in P@10= 0.18; $NDCG_{10} = 0.0985$); at the minimum threshold of 14 (with slightly lower values for the surrounding densities between 11 and 20) we get P@10 $= 0.32$ and $NDCG_{10} = 0.1823$, which is almost equal to the scores of the BTE al-
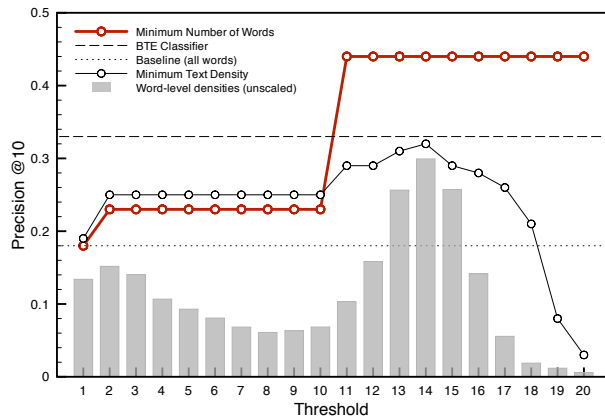

Figure 7: BLOGS06 Search Results

gorithm (P@10 $= 0.33$ and $NDCG_{10} = 0.1627$). For the simple sliding minimum *Number of Words*, we achieved a remarkable accuracy of P@10 $= 0.44$ and $NDCG_{10} = 0.2476$ for any minimum threshold between 11 and 100 words (an improvement by 144%/151% over the baseline and 33%/52% over BTE). We did not examine higher thresholds for practical reasons; at some point, of course, the precision would drop again because of lacking input. Figure 7 depicts the results for P@10; the $NDCG_{10}$ curves expose identical behaviour. In a direct comparison with the BLOGS06 competition, our results are of course somewhat lower since our strategy does not do opinion mining at all. However boilerplate removal seems to be strongly beneficial for this purpose: we can still compete with the lower 4 of the 16 contestants. One can therefore expect that the addition of our strategy to the opinion mining pipeline would further increase accuracy.

7. CONCLUSIONS AND FURTHER WORK

Conclusions. In this paper, we presented a simple, yet effective approach for boilerplate detection using shallow text features, which is theoretically grounded by stochastic text generation processes from Quantitative Linguistics.

We have shown that textual content on the Web can apparently be grouped into two classes, *long text* (most likely the actual content) and *short text* (most likely navigational boilerplate text) respectively. Through our systematical analysis we found that removing the words from the short text class alone already is a good strategy for cleaning boilerplate and that using a combination of multiple shallow text features achieves an almost perfect accuracy. To a large extent the detection of boilerplate text does not require any inter-document knowledge (frequency of text blocks, common page layout etc.) nor any training at token level.

We analyzed our boilerplate detection strategies on four representative multi-domain corpora (news, blogs and cross-domain) and also evaluated the impact of boilerplate removal for document retrieval. In all cases we achieve significant improvements over the baseline and accuracies that withstand and even outperform more complex competitive strategies, which incorporate inter-document information, n-grams or other heuristics.

The costs for detecting boilerplates are negligible, as it comes down simply to counting words.

Further Work. The presented results raise research issues in many different directions. Obviously, for boilerplate

detection we need to remove the "last 5%" of classification error. Even though the generality of our approach might suggest that it is universally applicable, we need to test our approach on other content domains and languages. Finally, we need to deeper investigate and extend our textual model from a Quantitative Linguistic perspective. To open up further possibilities, the algorithms used in this paper are available freely from the authors' website.[4]

8. REFERENCES

[1] G. Altmann. *Quantitative Linguistics - an international handbook*, chapter Diversification processes. de Gruyter, 2005.

[2] S. Baluja. Browsing on small screens: recasting web-page segmentation into an efficient machine learning framework. In *WWW '06: Proceedings of the 15th international conference on World Wide Web*, pages 33–42, New York, NY, USA, 2006. ACM.

[3] Z. Bar-Yossef and S. Rajagopalan. Template detection via data mining and its applications. In *WWW*, pages 580–591, 2002.

[4] M. Baroni, F. Chantree, A. Kilgarriff, and S. Sharoff. Cleaneval: a competition for cleaning web pages. In N. Calzolari, K. Choukri, B. Maegaard, J. Mariani, J. Odjik, S. Piperidis, and D. Tapias, editors, *Proceedings of the Sixth International Language Resources and Evaluation (LREC'08)*, 2008.

[5] D. Cai, S. Yu, J.-R. Wen, and W.-Y. Ma. Extracting content structure for web pages based on visual representation. In X. Zhou, Y. Zhang, and M. E. Orlowska, editors, *APWeb*, volume 2642 of *LNCS*, pages 406–417. Springer, 2003.

[6] D. Chakrabarti, R. Kumar, and K. Punera. Page-level template detection via isotonic smoothing. In *WWW '07: Proc. of the 16th int. conf. on World Wide Web*, pages 61–70, New York, NY, USA, 2007. ACM.

[7] D. Chakrabarti, R. Kumar, and K. Punera. A graph-theoretic approach to webpage segmentation. In *WWW '08: Proceeding of the 17th international conference on World Wide Web*, pages 377–386, New York, NY, USA, 2008. ACM.

[8] Y. Chen, W.-Y. Ma, and H.-J. Zhang. Detecting web page structure for adaptive viewing on small form factor devices. In *WWW '03: Proceedings of the 12th international conference on World Wide Web*, pages 225–233, New York, NY, USA, 2003. ACM.

[9] S. Debnath, P. Mitra, N. Pal, and C. L. Giles. Automatic identification of informative sections of web pages. *IEEE Trans. on Knowledge and Data Engineering*, 17(9):1233–1246, 2005.

[10] S. Evert. A lightweight and efficient tool for cleaning web pages. In N. Calzolari, K. Choukri, B. Maegaard, J. Mariani, J. Odjik, S. Piperidis, and D. Tapias, editors, *Proceedings of the Sixth International Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco, may 2008. European Language Resources Association (ELRA). http://www.lrec-conf.org/proceedings/lrec2008/.

[11] D. Fernandes, E. S. de Moura, B. Ribeiro-Neto, A. S. da Silva, and M. A. Gonçalves. Computing block importance for searching on web sites. In *CIKM '07*, pages 165–174, 2007.

[12] A. Ferraresi, E. Zanchetta, M. Baroni, and S. Bernardini. Introducing and evaluating ukwac, a very large web-derived corpus of english. In *Proceedings of the WAC4 Workshop at LREC 2008*.

[13] A. Finn, N. Kushmerick, and B. Smyth. Fact or fiction: Content classification for digital libraries. Joint DELOS-NSF Workshop on Personalisation and Recommender Systems in Digital Libraries (Dublin), 2001.

[14] D. Gibson, K. Punera, and A. Tomkins. The volume and evolution of web page templates. In *WWW'05*, pages 830–839, New York, NY, USA, 2005. ACM.

[15] J. Gibson, B. Wellner, and S. Lubar. Adaptive web-page content identification. In *WIDM '07: Proceedings of the 9th annual ACM international workshop on Web information and data management*, pages 105–112, New York, NY, USA, 2007. ACM.

[16] K. Hofmann and W. Weerkamp. Web corpus cleaning using content and structure. In *Building and Exploring Web Corpora*, pages 145–154. UCL Presses Universitaires de Louvain, September 2007.

[17] H.-Y. Kao, J.-M. Ho, and M.-S. Chen. Wisdom: Web intrapage informative structure mining based on document object model. *Knowledge and Data Engineering, IEEE Transactions on*, 17(5):614–627, May 2005.

[18] C. Kohlschütter. A densitometric analysis of web template content. In *WWW '09: Proc. of the 18th intl. conf. on World Wide Web*, New York, NY, USA, 2009. ACM.

[19] C. Kohlschütter and W. Nejdl. A Densitometric Approach to Web Page Segmentation. In *ACM 17th Conf. on Information and Knowledge Management (CIKM 2008)*, 2008.

[20] I. Ounis, C. Macdonald, M. de Rijke, G. Mishne, and I. Soboroff. Overview of the trec 2006 blog track. In E. M. Voorhees and L. P. Buckland, editors, *TREC*, volume Special Publication 500-272. National Institute of Standards and Technology (NIST), 2006.

[21] J. Pasternack and D. Roth. Extracting article text from the web with maximum subsequence segmentation. In *WWW '09: Proceedings of the 18th international conference on World wide web*, pages 971–980, New York, NY, USA, 2009. ACM.

[22] C. E. Shannon. A mathematical theory of communication. *Bell system techn. journal*, 27, 1948.

[23] M. Spousta, M. Marek, and P. Pecina. Victor: the web-page cleaning tool. In *WaC4*, 2008.

[24] K. Vieira, A. S. da Silva, N. Pinto, E. S. de Moura, a. M. B. C. Jo and J. Freire. A fast and robust method for web page template detection and removal. In *CIKM '06: Proc. 15th ACM int. conf. on Information and knowledge management*, pages 258–267, 2006.

[25] R. Vulanovic and R. Köhler. *Quantitative Linguistics - An international Handbook*, chapter Syntactic units and structures, pages 274–291. de Gruyter, 2005.

[26] L. Yi, B. Liu, and X. Li. Eliminating noisy information in web pages for data mining. In *KDD '03: Proc. of the 9th ACM SIGKDD int. conf. on Knowledge discovery and data mining*, pages 296–305, 2003.

---

[4] http://www.L3S.de/~kohlschuetter/boilerplate