A Densitometric Approach to Web Page Segmentation

Christian Kohlschütter
L3S / Leibniz Universität Hannover
Appelstr. 9a, 30167 Hannover
Germany
kohlschuetter@L3S.de

Wolfgang Nejdl
L3S / Leibniz Universität Hannover
Appelstr. 9a, 30167 Hannover
Germany
nejdl@L3S.de

ABSTRACT

Web Page segmentation is a crucial step for many applications in Information Retrieval, such as text classification, de-duplication and full-text search. In this paper we describe a new approach to segment HTML pages, building on methods from Quantitative Linguistics and strategies borrowed from the area of Computer Vision. We utilize the notion of text-density as a measure to identify the individual text segments of a web page, reducing the problem to solving a 1D-partitioning task. The distribution of segment-level text density seems to follow a negative hypergeometric distribution, described by Frumkina's Law. Our extensive evaluation confirms the validity and quality of our approach and its applicability to the Web.

Categories and Subject Descriptors

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

General Terms

Algorithms, Experimentation

Keywords

Web Page Segmentation, Full-text Extraction, Template Detection, Noise Removal

1. INTRODUCTION

Identifying and retrieving distinct information elements from the Web has increasingly become difficult. Besides the *main content* (e.g., an article) modern web pages also contain a bouquet of other textual elements such as navigation menus, user comments, text ads, snippet previews of related documents, legal disclaimers etc. Separating (segmenting) these distinct elements and eventually classifying them into relevant and non-relevant parts is essential for high-quality results. We consider three key application areas for web page segmentation: *(1) De-duplication.* Identical content information may be presented using different web page layouts. *(2) Content Extraction.* Besides the obvious benefits for Web-based news clipping etc., removing template noise might also increase classifier performance. *(3) Keyword-based Web search.* A page should be regarded less relevant to the query if the matched term only occurs in a template segment.

Until now, the segmentation problem has mainly been addressed by analyzing the DOM (Document Object Model) structure of an HTML page, either by rendering and visual analysis or by interpreting or learning the meaning and importance of tag structures in some way, both using heuristic as well as formalized, principled approaches. However, the number of possible DOM layout patterns is virtually infinite, which inescapably leads to errors when moving from training data to Web-scale. The actual retrievable unit – namely *text* – has only partially been investigated for the purpose of web page segmentation. Whereas it has been analyzed on the level of semantics and on term-level, a low-level pattern analysis is still missing.

Our Contributions. In this paper, we try to fill this gap as follows. (1) We define an abstract block-level page segmentation model which focuses on the low-level properties of text instead of DOM-structural information. (2) We concretize the abstract model. The key observation is that the number of tokens in a text fragment (or more precisely, its *token density)* is a valuable feature for segmentation decisions. This allows us to reduce the page segmentation problem to a 1D-partitioning problem. (3) We present the Block Fusion algorithm for identifying segments using the text density metric. (4) We present an empirical analysis of our algorithm and the block structure of web pages and evaluate the results, comparing with existing approaches.

Organization. In Section 2 we present the related work. Section 3 covers the problem discussion. Section 4 presents our simple, yet effective algorithm. In Section 5, we present our experimental results. Section 6 concludes with an outlook to future work.

2. RELATED WORK

Attempts to web page segmentation consider a variety of methods from different aspects. Most commonly, the structure of the web page (i.e., the DOM tree) is analyzed, in order to mine block-specific patterns, for example to separate and remove template elements from the actual main content. Bar-Yossef and Rajagopalan [4] identify template blocks by finding common shingles, similar to Gibson et al. [16] who also considers element frequencies for template detection. Debnath et al. compute an inverse block frequency for clas-

CIKM'08, October 26–30, 2008, Napa Valley, California, USA.

sification [13]. In [9], Chakrabarti et al. determine the "templateness" of DOM nodes by regularized isotonic regression. Yi et al. simplify the DOM structure by deriving a so-called Site Style Tree which is then used for classification [26]. Vieira et al. present an approach to template removal by identifying common DOM subtrees from a sample set and removing these structures from the whole collection [24]. Kao et al. separate blocks of DOM subtrees by comparing the entropies of the contained terms [20]. Vision-based approaches add information gained after rendering the DOM, such as Cai et al.'s VIPS algorithm [7], Chen et al.'s approach to tag pattern recognition [12] as well as Baluja's [3] method using decision tree learning and entropy reduction. Most recently, Chakrabarti et al. approached the webpage segmentation problem from a graph-theoretic perspective [10]. As shown by Cai et al. [8] and more recently by Fernandes et al. [15] the resulting segment structure can also be used for improving keyword-based search.

3. PROBLEM DISCUSSION

3.1 Segmentation as a Visual Problem

It is surprising how different the visual representation and the corresponding HTML document structure can be across different websites. Not only the use of different layouts contributes to this situation, but also the fact that there are versatile ways to model an identical layout, e.g. by varying between semantic and visual markup (`<EM>` vs. `<I>`), misusing `<TABLE>` structures for positioning non-tabular elements as well as completely neglecting HTML semantics for the layout. The latter has become very popular due to the use of CSS across most Web 2.0 websites, where tags usually are just `<DIV>` elements. This situation makes web page segmentation a non-trivial task. On Web-scale, rule-based or trained algorithms working on DOM-level are, due to the extreme heterogeneity of HTML style, susceptible to failure. On the other hand, vision-based approaches naturally have a higher complexity since the layout must be rendered (like in a browser) prior to analysis, which might be too slow to be incorporated into the Web crawling and indexing cycle.

Although we are examining the problem of web page segmentation from a textual perspective, there is a clear relationship to image segmentation from the field of Computer Vision: any DOM-level algorithm has to bear comparison with image recognition approaches, which span from k-means pixel clustering over histogram mode seeking and graph-partitioning to greedy region merging strategies [22]. In fact, we can draw a parallel from Shi's normalized cuts graph partitioning technique [22] to the recent work of Chakrabarti et al. [10], for instance. An example of a graph-independent approach is Haralick's and Shapiro's Region Growing [22]. Region Growing essentially is a greedy merging strategy; starting from one position in the image (e.g., top left corner), the grower iteratively fuses neighbored regions to larger ones (i.e., distinct pixels to sets of adjacent pixels). Under the assumption that the pixels are independent and uniformly distributed, the similarity of a region to another one is quantified by the deviation from the average *intensity* of that region – regions are merged if the deviation is insignificant. An application for Region Growing is text-block extraction from scanned newspaper images, where the algorithm is also known as Block Growing [2, 11].

3.2 Segmentation as a Linguistic Problem

In the field of Quantitative Linguistics, distributions of linguistic units such as words, syllables and sentences have been widely used as statistical measures to identify structural patterns in plain text documents, in particular for identifying subtopics [18] as well as for discovering changes of writing style [1] – both can be regarded as a special form of segmentation. In this discipline, it is a generally accepted assumption that the probability of a given class x in the corresponding unit's distribution solely is dependent on the probability of the neighboring lower class $x-1$ [17]:

$$P_x = g(x)\, P_{x-1} \tag{1}$$

For example, when a text is segmented into blocks of almost the same size, it is believed that the class distribution of term frequencies (occurrence probabilities) is negative hypergeometric (*Frumkina's law* or *law of text blocks*), which has been validated for various languages [6]:

$$P_x = \frac{\binom{-M}{x}\binom{-K+M}{n-x}}{\binom{-K}{n}}\ ,\ x = 0, 1, 2, ..., n \tag{2}$$

Taking this into account for segmentation, an obvious strategy is to examine the statistical properties of *subsequent* blocks with respect to their quantitative properties. In [18], for example, Hearst presents an algorithm which discovers sequences of adjacent paragraphs that belong to a topical unit within the document; which paragraphs get assigned to a particular subtopic is decided by a neighbored-block comparison based on term-frequency and cosine similarity.

Besides such an analysis of documents on the term-level, there are further interesting quantitative properties to consider. The distribution of document lengths follows the well-known Zipf distribution [14]. It might be reasonable to consider this distribution for segmenting intra-document text portions as well. The Zipf law states that the occurrence frequency of objects of a particular class is roughly inversely proportional to the corresponding rank of the class:

$$y = C\ x^{-b} \tag{3}$$

Another efficient quantum is *sentence length*. According to Altmann [1, 5], the creation of sentences is a stochastic process which follows a rhythm based on certain synergetic properties, i.e. the sentence lengths change along with the text flow. For analyzing changes in writing style he thus recommends not to compare random samples of a document but consecutive sentences instead. He concludes that also the occurrence probability of a particular sentence length x is a function of $x-1$ (yielding a hyperpascal distribution):

$$D_x = \frac{P_x - P_{x-1}}{P_x} \tag{4}$$

3.3 Segmentation as a Densitometric Problem

Coming back to the problem of *web page* segmentation, it is questionable whether the particular use of one specific HTML formatting style yields better signals for finding the "right" segmentation than another one. It is obvious that the *absence* of element tag information is a strong indicator

for the segmental unity of a text portion. We consider such text portions *atomic*. Could then perhaps the sheer *presence* of any element tag already be a sufficiently good signal for segmentation? While there are a few tags which separate by high chance (heading tags such as `<H1>`) and some which usually do not separate (the anchor text tag `<A>`), the majority of elements has unclear effects to segmentation. Thus, we may simply model a web page as a series of text portions (non-segmentable, *atomic blocks*) interleaved by a sequence of one or more opening or closing element tags, regardless of their meaning. We call such a sequence a *gap*. This simplifies the discussion to distinguishing the gaps which separate two segments and gaps which do not. Non-separating gaps may be discarded, resulting in larger text segments (*compound blocks*). While we can always *a priori* define certain tag-based rules for this decision finding, we focus on analyzing the blocks' inherent textual properties for this purpose.

Most likely, a segment gap is caused by a change in the text flow, e.g. from a list of short phrases (navigational text portions like "Home", "What's new", "Contact us") over a sequence of full sentences (for the main content) back to short phrases or one-sentence blocks for the page footer (e.g., "Copyright (c) 2008 by ... All rights reserved"). This setting is similar to the analysis of *writing style* by comparing sentence lengths (see Section 3.2). Due to the lack of proper sentences in template elements, it is difficult to define "sentence" in the web page scenario. Instead, we may substitute sentence length by *text density*, i.e. the number of words within a particular 2-dimensional area. Text density has been defined by Spool et al. in the field of Web Usability as the ratio between the total number of words in a block and the height of the rendered and printed block in inches [21]; a similar notion is known in Computer Vision, that is the *intensity* of an image region [22]. We transfer this concept to HTML text. The counterpart of a pixel in HTML is *character data* (the atomic text portion), an image region translates to a *sequence* of atomic text portions, which we also call *block* here. To determine a text block's "height", we word-wrap its text (not its rendered representation) at a constant line width $w_{\max}$ (in characters). The resulting block b_x's density $\rho(b_x)$ could then be formulated as follows:

$$\rho(b_x) = \frac{\text{Number of tokens in } b_x}{\text{Number of lines in } b_x} \tag{5}$$

This definition of text density has the elegant property that – except tokenization – no lexical or grammatical analysis needs to be performed. Given a proper wrapping width, it is supposed to serve as a discriminator between sentential text (high density) and template text (low density). We propose $w_{\max} = 80$. This is the traditional screen width of monospaced terminals and seems to fit the definitions of an English sentence: Assuming an average word length of 5.1 characters[1], we can write a maximum of $\frac{80}{5.1+1} = 13.1$ separate words (tokens) per line, which roughly covers one medium-sized sentence; obviously, the absolute maximum is 40 one-character tokens per line. It makes sense to exclude the last line of a multi-line block for the computation, since it would falsify the actual density when averaging if it is not completely filled to $w_{\max}$. Given the set of tokens T contained in the set of wrapped lines L covered by a block b_x, we can reformulate Equation 5 as follows.

[1]An overview of language-specific word lengths can be found at `http://blogamundo.net/lab/wordlengths/`

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

$$\rho(b_x) \quad = \quad \begin{cases} \frac{|T'(b_x)|}{|L(b_x)|-1} & |L(b_x)| > 1 \\ |\,T(b_x)\,| & \text{otherwise} \end{cases} \tag{6}$$

Now the density of a multi-line block is not influenced by the number of additional tokens (i.e., doubling the number of tokens leads to almost double the number of lines, which gets normalized again; see Equation 6). However, having only a few words (like "Contact us") still leads to a much lower density value, as expected.

While our text density measure does not consider lexical or grammatical properties of sentences at all, its role as a surrogate for sentence length may be well justified. Altmann [1] supports this by the rationale that *language* itself does actually not care about the existence or clear boundaries of particular lexical or grammatical units and that such units are rather an orthographical convention of the speech community. What seems more important than a proper definition of "sentence" is the measure of the units enclosed by the sentence (words, syllables, characters). The unit used for text density is the *token*, which basically is a variant of the (also diffuse) notion of "word"; in our case it is any contiguous sequence of non-whitespace characters, simplified to the set of contained literals and digits.

3.4 Segmentation as a 1-Dimensional Problem

The task of detecting block-separating gaps on a web page ultimately boils down to finding neighbored text portions with a significant change in the *slope* of the block-by-block text density. In Figure 1, we depict the desired segmentation[2] of the CIKM 2008 welcome page (`http://cikm2008.org/`), both visually as well as densitometrically. In the diagram, the density of the atomic text blocks is depicted as grey bars, HTML markup is indicated as white stripes and the expected segmentation boundaries are indicated as red vertical lines. Apparently, apart from the expected spikes, the distribution of text density appears to be a fairly good signal for textual similarity as well as for identifying full-text segments (block #5).

4. THE BLOCK FUSION ALGORITHM

As it turns out by the preceding discussion of the segmentation problem, we can essentially transfer parts from the perspective of Quantitative Linguistics as well as of Computer Vision to our setting. Due to Altmann's findings about the length dependence of neighbored sentences within the text flow and our corresponding findings on the text density, a greedy strategy seems a plausible algorithmic approach; besides being deterministic, an at least near-optimal result is likely. If we now indeed consider *text density* as being interrelated to the notion of *pixel intensity*, we may consider adopting the Block Growing strategy from image processing to. To avoid confusion with the pixel-based methods, we call this token-based method *Block Fusion*. The decision when to combine (fuse) two adjacent blocks now is made by comparing them with respect to their text densities instead

[2]Indisputably, there is no such thing like *the* segmentation, since segments may be considered at different granularities.

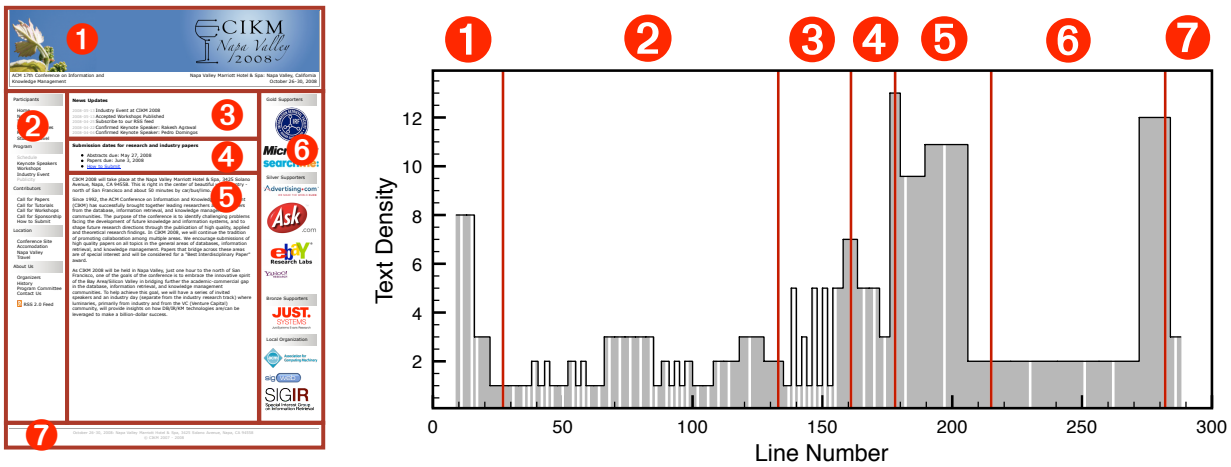


Figure 1: Visual vs. Densitometric Segmentation (expected results)

Algorithm 1 The Block Fusion algorithm (*plain*/*smoothed*)

Require: $B \Leftarrow$ The set of (initially atomic) blocks which partition the lines L

repeat
  loop $\leftarrow$ *false*
  for all $b_i \in B$ with $i > 1$ **do**
    if $\rho(b_{i-1}) = \rho(b_{i+1}) \wedge \rho(b_i) < \rho(b_{i-1})$ **then**
      ▷ Only checked for BF-SMOOTHED
      $b_{i+1} \leftarrow \{l \in L |\ l_{first}(b_{i-1}) \leq l \leq l_{last}(b_{i+1})\}$
      remove b_{i-1}
      remove b_i
      $i \leftarrow i + 1$ ▷ Skip b_{i+1}
      loop $\leftarrow$ *true*
    else if $\Delta\rho(b_{i-1}, b_i) \leq \vartheta_{\max}$ **then**
      $b_i \leftarrow \{l \in L |\ l_{first}(b_{i-1}) \leq l \leq l_{last}(b_i)\}$
      remove b_{i-1}
      loop $\leftarrow$ *true*
    end if
  end for
until loop = *false*

of pixel intensities. We may define this slope delta between two adjacent blocks x and y as:

$$\Delta\rho(x, y) = \frac{|\rho(x) - \rho(y)|}{\max(\rho(x), \rho(y))} \tag{7}$$

If the slope delta is below a certain threshold $\vartheta_{\max}$, we assume that the blocks belong to one segment and should therefore be fused. "To fuse" here means joining the lines of the two blocks x and y to a new block z, such that z spans from the first line of x to the last line of y. After this, x and y are replaced by z. As with the Block Growing strategy, we can iteratively continue with this operation until no pair of neighbored block exists which satisfies the threshold constraint.

In addition to that, we might also consider the following extension to this simple fusion strategy. As we can see from the example density distribution of the CIKM web page (Figure 1), there are some adjacent segments with alternating densities of 1.0/2.0/1.0, 1.0/5.0/1.0 etc. (this is the section about important dates – the dates are enclosed by `<SPAN>` tags, which create gaps). This may lead to high slope deltas close to 100% and therefore to less fusions than expected. We conclude that the surrounding blocks *dominate* the enclosed one. Our suggestion is to smooth these alternations by adding the following condition to the Block Fusion algorithm: if the text densities of the predecessor and successor of a block are identical and higher than its own density, all three blocks are fused. Of course, we will validate this heuristic against the plain strategy. See Algorithm 1 for a common representation of both strategies, BF-PLAIN and BF-SMOOTHED.

The computational complexity of Block-Fusion is trivial. Assuming we have N atomic blocks on a page, the cost per iteration is $c \cdot (N - 1)$ comparisons ($c = 1$ for BF-PLAIN, $c = 2$ for BF-SMOOTHED) and a maximum of $N - 1$ fusions per iteration occur. Because the iteration stops as soon as zero fusions occurred, the worst case that may occur is a single fusion per iteration (convergence is therefore guaranteed). The total number of operations for a maximum of k iterations until convergence therefore is:

$$(N - 1) + (N - 1 - 1) + \cdots + (N - k - 1) = O(N)$$

Two variables may influence the quality of the segmentation: the threshold $\vartheta_{\max}$ and the input blocks B. Regarding $\vartheta_{\max}$ we believe that this threshold is not document-specific but rather depends on the average style of the document class and its inherent quantitative properties. In Section 5.2 we determine an appropriate threshold value from a random sample of web documents. According to our definition of the simple block-gap model (Section 3.3), B describes the sequence of textual portions of the original HTML document. Whenever one or more opening or closing element tags are encountered, a new block is created, consisting of the plain text that is surrounded by markup; each block's text is initially word-wrapped by w_{max} characters (the wrapping does not change in the course of fusion).

Apart from special HTML tags whose nested character elements do not contribute to the text of the page (like `<SCRIPT>`, `<OPTION>` etc.) and the `<A>` tag, which we re-

gard as a core feature of hypertext markup and therefore do not consider a gap before or after this tag, we do not respect the element tag's semantic meaning or expected visual effect – a <H1> tag produces the same type of gap as a <B> tag, for example. Intuitively, we could of course claim that <H1> does indeed have a stronger impact on segmentation than a <B> tag, but this would again lead to heuristic, rule-based or DOM-structural approaches. For the evaluation, we will consider such a rule-based extension of our BF-SMOOTHED algorithm (which we call BF-RULEBASED for simplicity) which considers a set of specific *gap-enforcing* and *gap-avoiding* tags ($T_{ForceGap}$ and T_{NoGap}). Given the set of tags $T(x, y)$ between two segments x and y, to support this extension we have to change the slope delta function from $\Delta\rho(x, y)$ to $\Delta\rho'(x, y)$:

$$\Delta\rho'(x,y) = \begin{cases} +\infty & T(x,y) \cap T_{ForceGap} \neq \emptyset \\ -\infty & T(x,y) \subseteq T_{NoGap} \\ \Delta\rho(x,y) & \text{otherwise} \end{cases} \quad (8)$$

We consider the following gap-enforcing tags ($T_{ForceGap}$) as a good choice for the rule-based approach: H1-H6, UL, DL, OL, HR, TABLE, ADDRESS, HR, IMG, SCRIPT[3] (basically a subset of HTML block-level elements tags). For T_{NoGap}, we consider the following tags: A, B, BR, EM, FONT, I, S, SPAN, STRONG, SUB, SUP, U, TT (a subset of HTML inline element tags). When $\vartheta_{\max} = \infty$, this approach simply segments the document after every occurrence of a tag $\in T_{ForceGap}$ regardless of $\Delta\rho'$ or T_{NoGap} (Block Fusion has no effect in this case; we call this special variant JUSTRULES). Lower values of $\vartheta_{\max}$ represent a trade-off between markup-based and density-based segmentation. The examination of the effects of $\vartheta_{\max}$ are part of our evaluation.

5. EXPERIMENTAL EVALUATION

To demonstrate the stability and effectivity of the density-based Block Fusion strategy, we used two standard test collections: Webspam UK-2007[4] and the Lyrics dataset used in [10]. Despite its name the Webspam UK-2007 collection is a good snapshot of the U.K. Web, roughly consisting of 106 million pages from 115,000 hosts. Several hosts have already been classified as *spam/non-spam*. From this non-spam fragment (356,437 pages) we randomly picked 111 web pages coming from 102 different websites. We manually assessed these documents to define a comparable segmentation. These manual results were then compared against the following different clustering strategies:

1. WORDWRAP. Simply take all text of a page and wrap it after $w_{\max} = 80$ characters; every line is a segment.
2. TAGGAP. Every text portion between any tag (except A) is a segment.
3. BF-PLAIN, BF-SMOOTHED and BF-RULEBASED. As described in Section 4.
4. JUSTRULES. As described in Section 4.
5. GCUTS. As described in [10]. We did not implement this algorithm. Yet, a comparison of clustering performance scores is justified since both datasets comprise randomly chosen web pages and are of the same size.

[3]Occurrences of SCRIPT likely indicate a gap.
[4]http://www.yr-bcn.es/webspam/datasets/uk2007/

5.1 Statistical Properties of Web Page Text

First of all, we need to validate the assumptions on the actual quantitative linguistic properties of textual web page content. We assume that text density as defined in Equation 6 is a surrogate for sentence length. It should therefore also yield the same characteristic distribution (or at least one which satisfies Equation 1). To derive distinct classes i from the text density quotient of neighbored blocks, we use the following assignment in accordance with Eq. 4. Of note, adjacent blocks with the same density are regarded as one block, i.e. as a contiguous "sentence" which has been mistakenly separated:

$$X[i] = \left\lceil \frac{\Delta\rho(b_{x-1}, b_x)}{\rho(b_x)} \right\rceil \quad \forall \Delta\rho(b_{x-1}, b_x) \neq 0 \quad (9)$$

We used our manually created segmentation of the 111 web pages from the Webspam-UK2007 test collection and computed the class frequencies. Then we applied the Altmann-Fitter[5] to automatically determine one or more possible fits out of more than 200 supported discrete distributions for the given input data. The most significantly fitting probability distribution is negative hypergeometric (Equation 2 with $K = 2.30454$, $M = 0.10989$, $n = 17$), having $\chi^2 = 14.2394$, $P(\chi^2) = 0.3572$, $C = \chi^2 / \sum F(i) = 0.0061$, $d.f. = 13$ and is rated by the Altmann Fitter as a "very good fit". See Figure 2 for a graphical comparison; raw results are shown in Appendix A. While this differs from the initially assumed hyperpascal distribution, the general assumption (Equation 1) still holds and seems to abide by Frumkina's law. In fact, Vulanovic and Köhler assume [25] that Frumkina's law can be applied not only on term-level but to all types of linguistic units. We can show that this is at least the case for the distribution of text density quotients between adjacent blocks, coming to the conclusion that text density may indeed function as a surrogate for sentence length.

More, we also found that the distribution of the number of tokens in a segment abides by Zipf's law. This has already been shown on document-level [14], and it is just consistent to also find these properties on intra-page level. We were able to fit the distribution of segment-level word lengths of our manually segmented documents to $y = 1.086 \cdot x^{-0.7028}$, with $\chi^2 = 256.555$ and a root mean square error of 0.013.

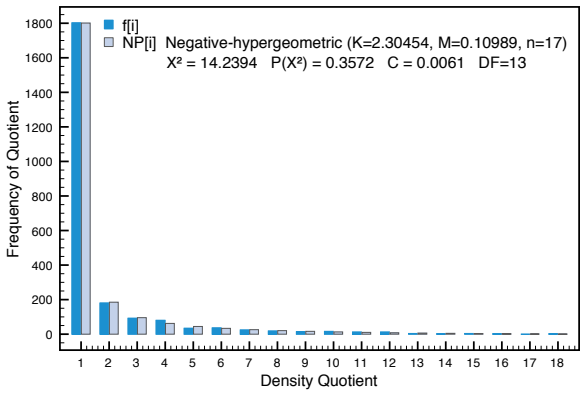


Figure 2: Probability Distribution of the Text Density Quotient of Adjacent Blocks (Equation 9)

[5]http://www.gabrielaltmann.de/

5.2 Segmentation Accuracy

Metrics. In order to quantify the accuracy of the segmentation computed by Block Fusion, we employ the two cluster correlation metrics *Adjusted Rand Index* (AdjRand) and *Normalized Mutual Information* (NMI) used in [10]. Both metrics determine the agreement between two clustering methods on a particular dataset, using a value between 0 (no agreement) to 1 (perfect agreement). The corresponding label vectors hold the information to which segment a particular token belongs to. AdjRand is Hubert's & Arabie's normalized extension to the Rand measure, which basically relates the number of agreements to the number of disagreements between the two given clusterings [19]. NMI measures the mutual dependence of the two solutions by relating their entropies. We use Strehl's & Ghosh's variant [23], which is defined as $NMI(X,Y) = \frac{I(X,Y)}{\sqrt{H(X)\ H(Y)}}$. Knowing that segmentation should follow Zipf's law on token-level, we can also measure and depict the consistency of a particular segmentation solution with this law. A deviation from the expected distribution is regarded a segmentation failure.

Results. We assume that for each variant of Block Fusion there is an optimal setting for the threshold $\vartheta_{\max}$ that is pre-determined by the underlying linguistic regularities. We probed Block Fusion using different settings for this threshold using our sample document set: for each candidate threshold, we compute the average AdjRand and NMI scores retrieved by a document-level comparison of the segmentations. The results are shown in Figures 3 and 4; we also plot the average number of resulting blocks for each setting as a reference. For BF-PLAIN and BF-SMOOTHED there seems to be an optimal threshold at $\vartheta_{\max} \approx 0.38$ for our sample document set, whereas any threshold between 0.3 and 0.4 seems reasonable. Starting with $\vartheta_{\max} = 0.4$, the accuracy decreases and finally drops dramatically with $\vartheta_{\max} \geqq 0.6$. See Figures 8, 9 and 10 for the corresponding visual and densitometric representation.[6]

We verified that the determined thresholds $\vartheta_{\max}$ are not particularly document-specific – we get almost the same optimal threshold for two random halves of the test set. For BF-RULEBASED the optimum is $\vartheta_{\max} \approx 0.6$. This means that the heuristically determined gap-enforcing tags do indeed contribute to the quality of segmentation, but the text densities do as well. The results for all applied clustering strategies are depicted in Table 1. Block Fusion clearly improves over WORDWRAP and TAGGAP. Interestingly, the scores of BF-PLAIN and BF-SMOOTHED are almost identical to GCUTS [10], which is a surprising achievement for a markup-agnostic approach. At last, BF-RULEBASED in fact outperforms any other approach. While it is close to the quality of JUSTRULES (whose accuracy confirms the effectiveness of our heuristic segmentation rules for the evaluated dataset), it also shows that our heuristics were not perfect and Block Fusion was able to improve them. We also examined the impact of $w_{\max}$ to the accuracy (see Figure 5). It appears that this word-wrap boundary is stable for widths between 80 and 110. This confirms the assumption on the relation between language-specific average sentence length and line width. Theoretically, we could optimize it to $w_{\max} = 90$, but this would only increase accuracy by less than 0.01 on average.

Finally, Figure 6 shows a log-log plot of block-level tokens counts for all considered algorithms (except GCUTS). All Block Fusion-based approaches as well as JUSTRULES and the manual segmentation expose the typical straight line known from Zipf distributions. As expected, TAGGAP and WORDWRAP obviously do not show this behavior. This means that Block Fusion is indeed able to transform the tag-induced segmentation to a segmentation which resembles the same statistical properties as the expected ones.

5.3 Performance

Since Block Fusion is designed as an iterative algorithm, we should consider the iteration behavior in terms of *average accuracy error* $(1 - accuracy)$ – we expect this error to monotonously decrease per iteration, just as the number of remaining blocks. Figure 7 reveals that most of the error gets removed already after the first iteration. Even though on average more blocks are fused during the following iterations, these fusions do not contribute to improving accuracy. Block Fusion achieves this performance because it can fuse an arbitrary number of preceding atomic or compound blocks with similar density in one iteration (see Algorithm 1). Notably, for the used test data, the total processing time per page was only 15ms on a standard laptop.

5.4 Application to Near-Duplicate Detection

Setup. Finally we quantify the usefulness of our segmentation for the purpose of near-duplicate detection. Again we compare the results from Block Fusion against [10], where the LYRICS dataset was used to evaluate the accuracy of detecting web pages with the same content but different appearance. The dataset consisted of 2359 web pages song lyrics by six popular artists (ABBA, Beatles, BeeGees, Bon Jovi, Rolling Stones and Madonna), taken from the three websites `absolutelyrics.com`, `seeklyrics.com` and `lyricsondemand.com`. The six artists were deliberately chosen to minimize the effect of false-positives on the evaluation caused by cover songs. As we were unable to acquire the original dataset, we crawled the three websites again using the same setup, resulting in 6982 web pages (which is likely to be a superset of the initial crawl by Chakrabarti et al.). By matching artist and title, 1082 songs have been determined that appear on all three websites (i.e., on 3246 web pages). In addition to that, 3246 other web pages have been randomly chosen from the three websites (1082 for each). This setup allows a relatively clean comparison between the true-positive and true-negative rates of a de-duplication algorithm. To determine what a near-duplicate is and what is not, the same heuristic was used as in [10]: For each page of a pair of candidate pages, the tokens of the largest text segment are used to create 8 shingle fingerprints using the min-hash algorithm, with a window of 6 tokens. A pair of pages is regarded a near-duplicate if the pages share at least 50% of the shingles. The largest text segment simply is determined by counting the number of enclosed tokens; in our setup, segments containing at least 50% hyperlinked textual content are discarded since they are likely not to contain the main content despite their length.

[6]The short segments seen in Figure 9 could not be fused by BF-Smoothed, because the smoothening criterion $\rho(b_{i-1}) = \rho(b_{i+1})$ was not met. We heuristically found the improved criterion $\rho(b_{i-1}) \leqq 5 \wedge \rho(b_{i+1}) \leqq 5$ which indeed fuses the segments correctly, while improving the accuracy scores only by ca. 0.02. We therefore consider this improvement as insignificant and omit it from the proposed solution.

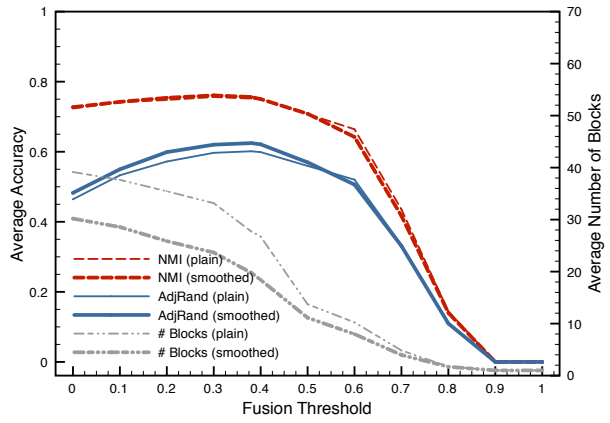


Figure 3: Optimizing $\vartheta_{\max}$ (BF-plain/smoothed)

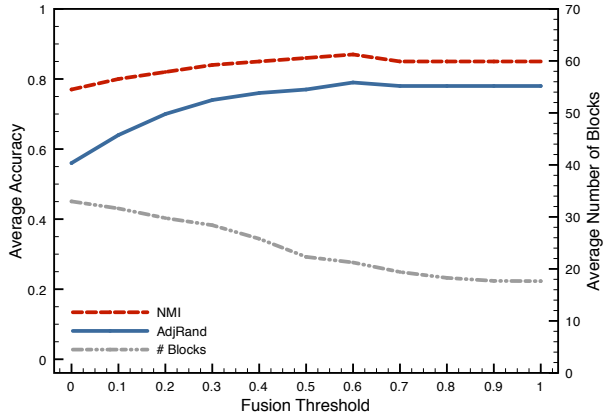


Figure 4: Optimizing $\vartheta_{\max}$ (BF-rulebased)

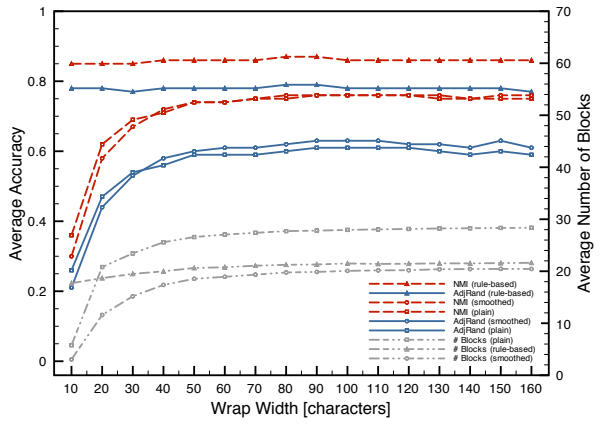


Figure 5: Impact of $w_{\max}$ on Average Accuracy

	AdjRand	NMI	# Blocks
WordWrap	0.25	0.59	25.0
TagGap	0.43	0.65	69.43
Bf-plain	**0.60**	**0.75**	27.72
Bf-smoothed	**0.62**	**0.76**	19.77
Bf-rulebased	**0.79**	**0.87**	21.24
JustRules	0.78	0.84	17.64
(GCuts)	(0.60)	(0.76)	-

Table 1: Achieved average Accuracies

	C	b	χ^2	Error
Bf-plain	1.04024	0.74899	98.15	0.00438
Bf-smoothed	1.03643	0.73334	87.38	0.00538
Bf-rulebased	1.47937	0.67028	649.36	0.02096
JustRules	1.08526	0.70280	256.56	0.01372

Table 2: Zipf Distribution Parameters

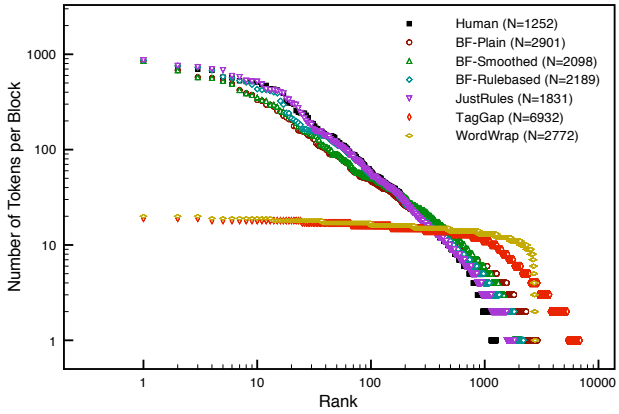


Figure 6: Validation of Zipf Law on Block Level

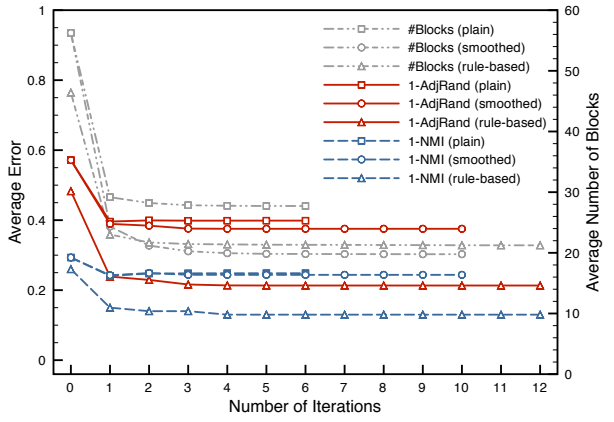


Figure 7: Iteration Behavior

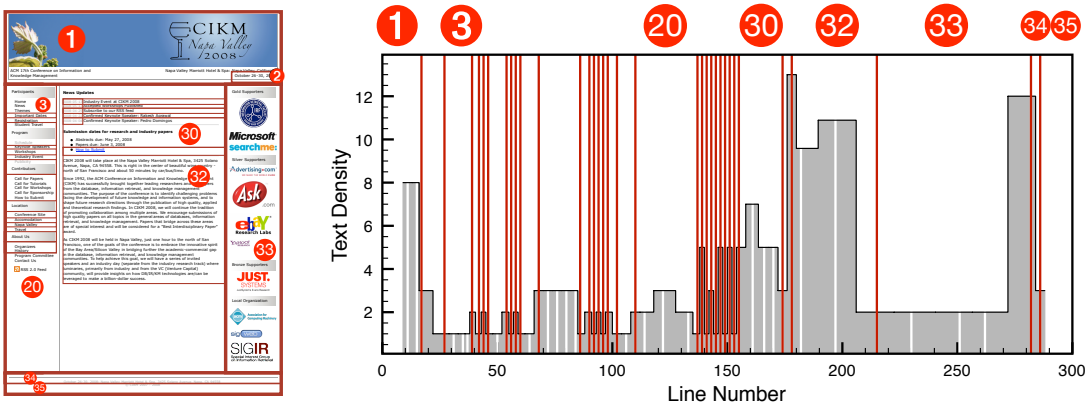
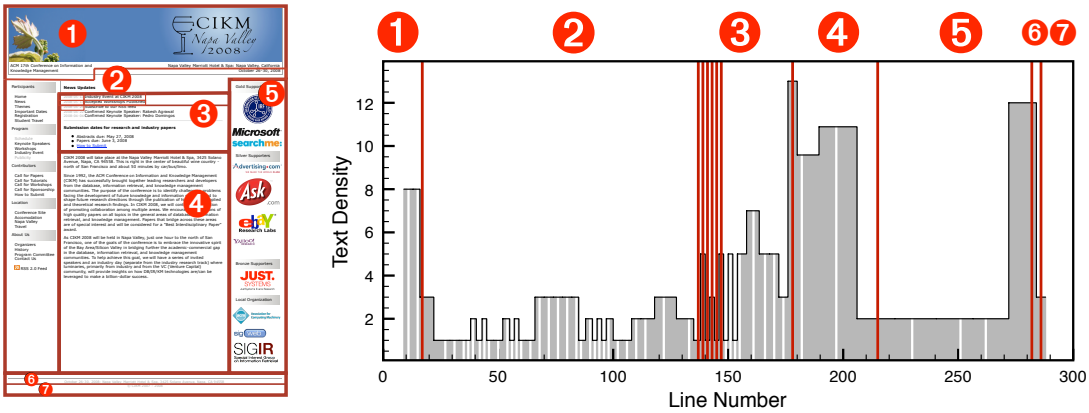


Figure 8: Visual vs. Densitometric Representation of the Segmentation (retrieved by BF-plain)



See Footnote 6 for an explanation of the short segments between lines 131 and 147.

Figure 9: Visual vs. Densitometric Representation of the Segmentation (retrieved by BF-smoothed)

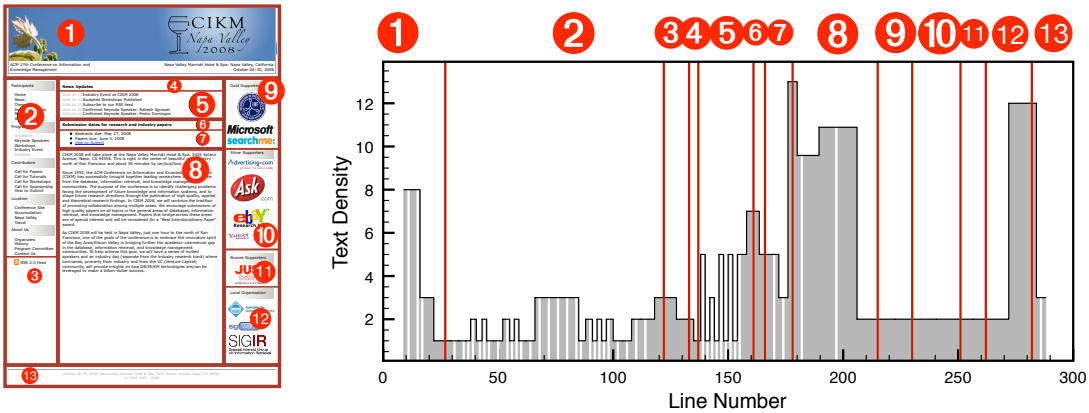


Figure 10: Visual vs. Densitometric Representation of the Segmentation (retrieved by BF-rulebased)

Results. The resulting true positive/negative scores corresponding to each algorithm (including a comparison to the text as a whole, FullText) are shown in Table 3. JustRules is the narrow winner with respect to finding duplicates, but all Block Fusion variants perform equally well for detecting non-duplicates and significantly perform better than GCuts, even the simplest variant BF-plain.

	True Duplicate Pairs	True Non-Duplicate Pairs
Total	3246	3246
FullText	19.9%	96.3%
WordWrap ($w_{\max} = 80$)	5.4%	76%
TagGap	16.9%	88.5%
Bf-plain ($\vartheta_{max} = 0.38$)	72.2%	100%
Bf-smoothed ($\vartheta_{max} = 0.38$)	73.1%	100%
Bf-rulebased ($\vartheta_{max} = 0.6$)	86.3%	100%
JustRules	89.4%	100%
(GCuts)	(61.7%)	(99.9%)

Table 3: Duplicate Detection Accuracy

6. CONCLUSIONS AND FUTURE WORK

Conclusions. The problem of web page segmentation can be seen from a quantitative linguistic point of view as a problem of identifying significant changes of particular statistical properties within the considered text. As we demonstrate in this paper, an effective property is *token-level text density*, which can be derived from vision-based measures. This text density follows the same fundamental linguistic law (Frumkina's Law) as many other linguistic units. In addition to that, the distribution of the expected number of tokens in a segment follows Zipf's law. Our new algorithm for web page segmentation, built upon the region growing strategy known in Computer Vision, performs significantly better than the state-of-the-art graph-theoretic algorithm, as our experimental evaluation on large real-world data sets demonstrates.

Outlook and Future Work. The approach presented in this paper is orthogonal to existing work and considers new and complementary aspects to solve the segmentation task. As shown by the rule-based Block Fusion hybrid, a more sophisticated combination of other strategies and the Block Fusion algorithm promises further improved segmentation quality. Since the considered linguistic properties seem to be mostly language-independent, the next logical step is to evaluate these findings on a multilingual corpus. In particular, we need to discuss the influence of the wrapping parameter $w_{\max}$ and threshold $\vartheta_{\max}$ on different languages. Further work should also find an explanation of the discovered statistical behavior from a purely linguistic perspective. We also want to investigate the use of our techniques in other areas of Information Retrieval, including block-level ranking, block-level link analysis and block-level classification.

Acknowledgments

We would like to express our gratitude to Professor Dr. Gabriel Altmann for providing the Altmann Fitter software and for his good advice on details of discrete distribution functions used in Quantitative Linguistics.

7. REFERENCES

[1] Gabriel Altmann. Verteilungen der Satzlängen (Distribution of Sentence Lengths). In K.-P. Schulz, editor, *Glottometrika 9*. Brockmeyer, 1988.

[2] A. Antonacopoulos, B. Gatos, and D. Bridson. Page segmentation competition. *Document Analysis and Recognition, 2007. ICDAR 2007. Ninth International Conference on*, 2:1279–1283, 23-26 Sept. 2007.

[3] Shumeet 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.

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

[5] Karl-Heinz Best. *Quantitative Linguistics - An International Handbook*, chapter Satzlänge (Sentence length), pages 298–304. de Gruyter, 2005.

[6] Karl-Heinz Best. Sprachliche Einheiten in Textblöcken. In *Glottometrics 9*, pages 1–12. RAM Verlag, Lüdenscheid, 2005.

[7] Deng Cai, Shipeng Yu, Ji-Rong Wen, and Wei-Ying 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.

[8] Deng Cai, Shipeng Yu, Ji-Rong Wen, and Wei-Ying Ma. Block-based web search. In *SIGIR '04: Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 456–463, New York, NY, USA, 2004. ACM.

[9] Deepayan Chakrabarti, Ravi Kumar, and Kunal Punera. Page-level template detection via isotonic smoothing. In *WWW '07: Proceedings of the 16th international conference on World Wide Web*, pages 61–70, New York, NY, USA, 2007. ACM.

[10] Deepayan Chakrabarti, Ravi Kumar, and Kunal 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.

[11] Ming Chen, Xiaoqing Ding, and Jian Liang. Analysis, understanding and representation of chinese newspaper with complex layout. *Image Processing, 2000. Proceedings. 2000 International Conference on*, 2:590–593 vol.2, 2000.

[12] Yu Chen, Wei-Ying Ma, and Hong-Jiang 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.

[13] Sandip Debnath, Prasenjit Mitra, Nirmal Pal, and C. Lee Giles. Automatic identification of informative sections of web pages. *IEEE Trans. on Knowledge and Data Engineering*, 17(9):1233–1246, 2005.

[14] Lukasz Debowski. Zipf's law against the text size: a half-rational model. In *Glottometrics 4*, pages 49–60. RAM Verlag, Lüdenscheid, 2002.

[15] David Fernandes, Edleno S. de Moura, Berthier

Ribeiro-Neto, Altigran S. da Silva, and Marcos André Gonçalves. Computing block importance for searching on web sites. In *CIKM '07: Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pages 165–174, New York, NY, USA, 2007. ACM.

[16] David Gibson, Kunal Punera, and Andrew Tomkins. The volume and evolution of web page templates. In Allan Ellis and Tatsuya Hagino, editors, *WWW (Special interest track)*, pages 830–839. ACM, 2005.

[17] Peter Grzybek. On the systematic and system-based study of grapheme frequencies - a re-analysis of german letter frequencies. In G. Altmann, K.-H. Best, and P. Grzybek et al., editors, *Glottometrics 15*, pages 82–91. RAM Verlag, Lüdenscheid, 2007.

[18] Marti A. Hearst. Multi-paragraph segmentation of expository text. In *Proceedings of the 32nd annual meeting on Association for Computational Linguistics*, pages 9–16, Morristown, NJ, USA, 1994. Association for Computational Linguistics.

[19] Lawrence Hubert and Phipps Arabie. Comparing partitions. *Journal of Classification*, 2(1):193–218, December 1985.

[20] Hung-Yu Kao, Jan-Ming Ho, and Ming-Syan 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.

[21] Jared M. Spool, Tara Scanlon, Carolyn Snyder, Will Schroeder, and Terri DeAngelo. *Web site usability: a designer's guide*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1999.

[22] George Stockman and Linda G. Shapiro. *Computer Vision*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2001.

[23] Alexander Strehl and Joydeep Ghosh. Cluster ensembles — a knowledge reuse framework for combining multiple partitions. *J. Mach. Learn. Res.*, 3:583–617, 2003.

[24] Karane Vieira, Altigran S. da Silva, Nick Pinto, Edleno S. de Moura, ao M. B. Cavalcanti Jo and Juliana Freire. 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*, pages 258–267, New York, NY, USA, 2006. ACM.

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

[26] Lan Yi, Bing Liu, and Xiaoli Li. Eliminating noisy information in web pages for data mining. In *KDD '03: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 296–305, New York, NY, USA, 2003. ACM.

APPENDIX

A. OUTPUT FROM ALTMANN-FITTER

```
-- ALTMANN-FITTER 2.1 --
Result of fitting

Input data: hist-1.dat
Distribution: Negative hypergeomet-
ric (K,M,n)
Sample size: 2334

Moments:
M1 =  1.8106
M2 =  4.4963
M3 = 34.2793
M4 =356.4193

Best method is
Method 1 of 2

Parameters:
K = 2.30453585151999
M = 0.109889153462268
n = 17

DF =13
χ² = 14.2394  P(χ²) =  0.3572  C =  0.0061
```

X[i]	F[i]	NP[i]
1	1802	1800.9989
2	180	184.9156
3	92	95.4882
4	80	62.2025
5	34	44.5585
6	36	33.5436
7	24	25.9808
8	18	20.4556
9	14	16.2396
10	16	12.9186
11	12	10.2396
12	12	8.0391
13	4	6.2069
14	2	4.6669
15	4	3.3651
16	2	2.2639
17	0	1.3385
18	2	0.5779