Content Code Blurring: A New Approach to Content Extraction

Thomas Gottron
Institut für Informatik
Johannes Gutenberg-Universität Mainz
55099 Mainz, Germany

E-mail: gottron@uni-mainz.de

Abstract

Most HTML documents on the World Wide Web contain far more than the article or text which forms their main content. Navigation menus, functional and design elements or commercial banners are typical examples of additional contents. Content Extraction is the process of identifying the main content and/or removing the additional contents. We introduce content code blurring*, a novel Content Extraction algorithm. As the main text content is typically a long, homogeneously formatted region in a web document, the aim is to identify exactly these regions in an iterative process. Comparing its performance with existing Content Extraction solutions we show that for most documents content code blurring delivers the best results.*

1. Introduction

Nowadays most HTML documents on the World Wide Web are generated from templates by content management systems. In addition to the main textual content they comprise several additional contents, e.g. navigation menus, functional and design elements or commercials. Already in 2005 Gibson, Punera and Tomkins [3] estimated those additional contents to account for around 40 to 50% of the data on the web, predicting this ratio to increase constantly.

Content Extraction (CE) is the process of determining those parts of an HTML document which represent its main textual content. Several applications benefit from CE under different aspects: Web Mining and Information Retrieval (IR) applications use CE to pre-process the raw HTML data to reduce noise and to obtain more accurate results, other applications use CE to reduce the document size for presentation on screen readers and small screen devices.

This paper introduces *content code blurring*, a novel CE algorithm. It is robust to invalid or badly formatted HTML documents, it is fast and concerning its extraction performance delivers very good results on most documents.

We proceed as follows: in section 2 we give an overview of related work in the field of CE, mentioning in particular the algorithms we will use for comparison during evaluation. The content code blurring algorithm itself is defined and explained in section 3. We describe our evaluation setup and compare the performance of our approach with other CE methods in 4 before concluding the paper with a discussion of the results and proposals for further work.

2. Related work

One of the more prominent solutions for CE is the Crunch framework [5] of Gupta et al. Crunch combines several heuristics to discover and remove different kinds of additional content. The main objective of Crunch is to restructure and optimise HTML documents for presentation on small screen devices or screen readers. To detect link lists, Crunch uses a link quota filter (LQF) similar to the one developed by Mantratzis et al. in [6]. LQF uses a high ratio of link content in document blocks to detect navigation menus and similar structures. Debnath et al. introduced the FeatureExtractor algorithm and its extension the K-FeatureExtractor in [1]. They segment a web document into blocks and analyse them for the prevalence of particular features like text, images, JavaScript etc. The extraction process is based on retrieving the blocks which correspond best to a desired feature, e.g. text for the main content of a news article. Finn et al. described the Body Text Extraction (BTE) algorithm in [2] as a pre-processor for their application classifying news articles on the web. BTE identifies a continuous part of the document which contains most of the text while excluding most of the tags. Pinto et al. [7] extended the BTE approach to construct Document Slope Curves (DSC). They use a windowing technique to locate also distributed and interrupted parts of the document which fit the main content characteristics as formulated for BTE.

Rahman et al. [8] list requirements a CE system for

HTML documents should comply with. Being generic enough to work with any website and using a fast extraction algorithm are the most important aspects they mentioned. In [4] we developed a way to measure, evaluate and compare CE algorithms based on the common IR measures precision, recall and F1. In the course of this work, we also compared different CE approaches and found an adaptation of DSC to be the best performing CE algorithm.

3. Content Code Blurring

The idea underlying content code blurring is to take advantage of typical visual features of the main and the additional contents. Additional contents are usually highly formatted and contain little and short texts. The main text content, on the other hand, is commonly long and homogeneously formatted. The example document in figure 1 demonstrates this observation.

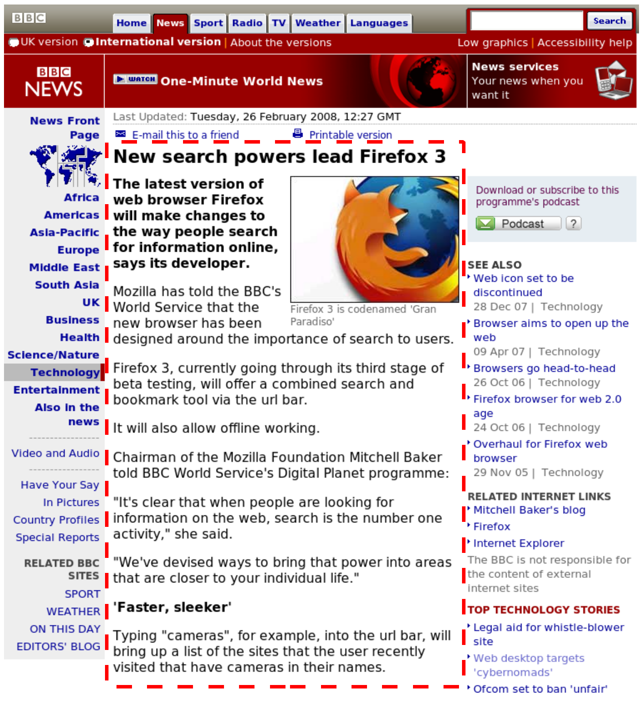
As in the source code of an HTML document any change of format is indicated by a tag, we will try to identify those parts of the document which contain a lot of text and few or no tags.

3.1. Concept and Idea

So, the idea and aim of content code blurring is to locate regions in a document which contain mainly content and little code. To formalise this task we need to define what we mean by content and by code and how to measure their amount in a document region.

The question of what is content and what is code can be answered quite easily. The tags in the source code correspond to code as they provide the structure, layout and formatting of a web document. The text outside the tags, instead, contributes the content. We use this basic distinction to obtain a suitable document representation in two different ways. The first approach strikes a new path for document representations in the CE context by determining for each single character whether it is content or code. Thus, a document is interpreted as a sequence of code and content characters. The second approach is based on a token sequence as used by BTE and DSC. Each tag and each word correspond to a token. The whole document is accordingly represented as a sequence of tag and word tokens. Both ways lead to a representation of a document as a sequence of atomic elements which are either content or code. We will refer to this vector from now on as the *content code vector* (CCV). This document representation is also very robust to syntax errors in the HTML code: as long as the tags can be identified it is possible to build a CCV.

For each single element in the CCV we determine a ratio of content to code in its vicinity to find out if it is surrounded mainly by content or by code. If for several elements in a row this *content code ratio* (CCR) is high, i.e. they are surrounded mainly by text and only by a few tags, we have found a region of the document with a relatively uniform format. So, a region with high CCR values corresponds to our idea of a region containing the main content. To calculate the CCR we use a process inspired by the blurring filters of image processing applications.



Last Updated: Tuesday, 26 February 2008, 12:27 GMT

New search powers lead Firefox 3

The latest version of web browser Firefox will make changes to the way people search for information online, says its developer.

Firefox 3 is codenamed 'Gran Paradiso'

Mozilla has told the BBC's World Service that the new browser has been designed around the importance of search to users.

Firefox 3, currently going through its third stage of beta testing, will offer a combined search and bookmark tool via the url bar.

It will also allow offline working.

Chairman of the Mozilla Foundation Mitchell Baker told BBC World Service's Digital Planet programme:

"It's clear that when people are looking for information on the web, search is the number one activity," she said.

"We've devised ways to bring that power into areas that are closer to your individual life."

'Faster, sleeker'

Typing "cameras", for example, into the url bar, will bring up a list of the sites that the user recently visited that have cameras in their names.

Download or subscribe to this programme's podcast

SEE ALSO
- Web icon set to be discontinued
  28 Dec 07 | Technology
- Browser aims to open up the web
  09 Apr 07 | Technology
- Browsers go head-to-head
  26 Oct 06 | Technology
- Firefox browser for web 2.0 age
  24 Oct 06 | Technology
- Overhaul for Firefox web browser
  29 Nov 05 | Technology

RELATED INTERNET LINKS
- Mitchell Baker's blog
- Firefox
- Internet Explorer

The BBC is not responsible for the content of external internet sites

TOP TECHNOLOGY STORIES
- Legal aid for whistle-blower site
- Web desktop targets 'cybernomads'
- Ofcom set to ban 'unfair'

Figure 1. An example for a web document with an outlined main content. The main text content is long and homogeneously formatted, while additional contents, like the navigation menu, commercials or link lists are highly structured and contain short texts.

3.2. Blurring the Content Code Vector

For the CCR calculation we represent the CCV as a vector of floating point values. Each entry in the vector is initialised with a value of 1 if the according element is of type content and with a value of 0 for code.

To obtain the CCR we calculate for each entry a weighted and local average of the values in a neighbourhood with a fixed symmetric range. If all the elements in this neighbourhood started with a value of 1, also the average will be 1; the same is valid for neighbourhoods with an initial value of 0. In inhomogeneous neighbourhoods the average value will be between 0 and 1 and depends on the values of the surrounding elements. If they are mainly content, the ratio will be high, if they are mainly code, the ratio will be low. So, the average values have exactly the properties we need for our CCR values.

The weights in the average calculation are used to model a stronger influence of near elements and a weaker influence for those further away. We chose the weights according to a Gauss distribution to obtain this effect. To further realise an influence of elements which are beyond the neighbourhood boundaries we iteratively repeat this process of calculating neighbourhood averages. In each iteration we use the resulting vector of averages as input for the next step. The iteration is stopped as soon as the values start to settle.

Visually the whole process corresponds to constructing a one dimensional image from the atomic elements, in which each pixel represents a single element and is initially coloured white if it represents content and black if it represents code. The calculation of weighted neighbourhood averages corresponds to applying a Gaussian blurring filter[1] – hence the name content code *blurring*.

Figure 2 demonstrates this visual interpretation. The first image has been generated from HTML source code as described above. When blurring the image the abrupt transitions between black and white are smoothed into shades of grey. The parts of the image which were initially mainly black end up in darker shades, those which have initially been mainly white will remain in brighter shades. Repeating the process shows a convergence of the shades. Translated into CCR this means: the bright areas have a high ratio of content to code and are accordingly rich in content, the dark ones have a low ratio and are rich in code.

3.3. Implementation and Adaptations

To find those regions in a document which contain mainly content corresponds to selecting those elements of the CCV which have a high CCR value, i.e. a value closer to 1. Accordingly, an element in the CCV is considered to be part of the main content, if it has a CCR value above a fixed threshold t. Furthermore, in the character based versions we extract words always entirely, even if only some of their characters are above the threshold.

In [4] we observed that most CE algorithms have problems with highly fragmented contents, e.g. due to a lot of in-text hyperlinks. This caused all CE methods to perform remarkably poor on wiki style web documents. A high rate of in-text links is problematic also for our attempt to find areas with few tags. Hence, we will analyse a variation of the algorithm which is intended to cope with this problem. Thinking of our initial idea of finding text blocks with a uniform layout we create an *adapted CCB* (ACCB) which ignores anchor-tags entirely during the creation of the CCV. After all, hyperlinks are not influencing the format intentionally. Their visual influence is more a side-effect of the necessity to reference another document.

[1]Gaussian filters can be found in nearly all image processing programs. They achieve the blurring effect in an image by spreading the colour value of a pixel to its neighbour pixels according to a Gauss distribution.

Table 1. Overview of the evaluation packages.

Package	URL	Size
bbc	http://news.bbc.co.uk	1000
chip	http://www.chip.de	361
economist	http://www.economist.com	250
espresso	http://espresso.repubblica.it	139
golem	http://golem.de	1000
heise	http://www.heise.de	1000
manual	several	65
repubblica	http://www.repubblica.it	1000
slashdot	http://slashdot.org	364
spiegel	http://www.spiegel.de	1000
telepolis	http://www.telepolis.de	1000
wiki	http://de.wikipedia.org	1000
yahoo	http://news.yahoo.com	1000
zdf	http://www.heute.de	422

This approach might be too specialised for wiki style pages, though, and even counterproductive for other HTML documents. LQF, for example, uses the presence of hyperlinks as a sure sign for additional contents. Ignoring hyperlinks might accordingly weaken the general extraction performance. Hence, we will particularly compare the performance of ACCB with the original algorithm.

So, we have three variations of content code blurring to evaluate: the character based version in its original form (CCB), with the adaptation of ignoring hyperlinks (ACCB) and the token based version (TCCB).

4. Evaluation

For evaluation, we will use the same evaluation methodology as in [4]. We provide a gold standard for the main text content of web documents. To compare the extract provided by a CE algorithm with the gold standard we need to compute an overlap between the texts. For this purpose we determine the longest common (not necessarily continuous) subsequence of words in both texts. Considering this subsequence as the intersection between retrieved (i.e. extracted) and relevant text (i.e. part of the gold standard) allows to apply IR measures like recall, precision and F1.

The evaluation data is organised in packages which are listed in table 1. The manual package consists of documents for which the main content has been outlined manually. For the other packages we applied dedicated programs, which are capable of harvesting the main content from the documents of particular web sites. In this way we obtained large amounts of documents for large scale tests: altogether

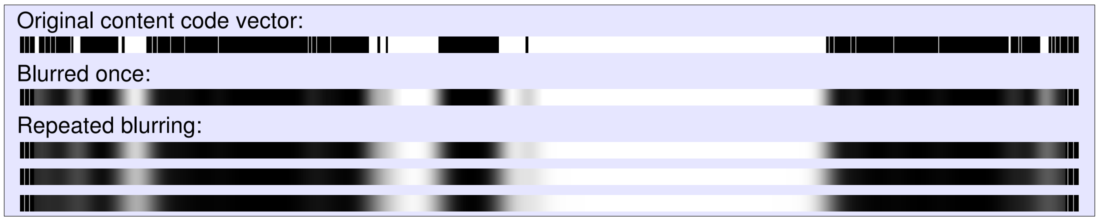


Figure 2. Blurring a content code vector interpreted as a greyscale image.

14 packages with a total of 9,601 documents. These documents cover different document styles, layout techniques and main content lengths.

In most documents the main content can quite easily be agreed upon by human users. The slashdot package is the only problematic case. Here a short news article is always extended by a long discussion thread of slashdot users. We considered this discussion not to be part of the main content.

4.1. Fixing the Parameters

The content code blurring algorithms have two main parameters: the influence range and the threshold value t. The range defines the direct influence of the atomic elements on their neighbourhood; the threshold provides the minimum CCR value an element has to satisfy for being declared part of the main content. To find good settings for these parameters we evaluated the performance of the CCB, ACCB and TCCB manually on several test documents.

We found quite soon that setting $t = 0.75$ performs well for all influence ranges and all variations of the algorithm. Keeping the threshold fixed allowed an easy exploration of settings for the range. A range of 40 turned out to be a good choice for the character based CCB and ACCB algorithms, while for the token based TCCB a range setting of 25 is the best choice.

4.2. Results

The average F1 results for extracting the main content from the evaluation documents are shown in table 2. The table also includes the performance of DSC, Crunch and LQF. DSC is of particular interest, as it was the best algorithm in our last comparison. The alternative of not using any CE is listed as "Plain" method and forms the baseline. Each CE algorithm should perform better than not using CE.

The results of the content code blurring algorithms are generally quite good. First of all, we can notice that except for the wiki pages all versions achieve better results than the "Plain" baseline. This qualifies them as valid CE methods. The second important insight is that ACCB does not show a significant drawback in comparison to CCB. So, the adaptation of ignoring hyperlink tags during the construction of

Table 2. Evaluation results of new single document algorithms: average F1.

	ACCB	CCB	TCCB	DSC	Crunch	LQF	Plain
bbc	0.924	0.923	0.914	0.937	0.756	0.826	0.595
chip	0.703	0.716	0.842	0.708	0.342	0.502	0.173
economist	0.890	0.914	0.903	0.881	0.815	0.720	0.613
espresso	0.875	0.876	0.871	0.862	0.810	0.666	0.624
golem	0.959	0.939	0.947	0.958	0.837	0.806	0.502
heise	0.916	0.841	0.821	0.877	0.810	0.787	0.575
manual	0.419	0.420	0.404	0.403	0.382	0.381	0.371
repubblica	0.968	0.964	0.918	0.925	0.887	0.816	0.704
slashdot	0.177	0.160	0.269	0.252	0.123	0.127	0.106
spiegel	0.861	0.858	0.910	0.902	0.706	0.775	0.549
telepolis	0.908	0.913	0.902	0.859	0.910	0.906	0.858
wiki	0.682	0.403	0.660	0.594	0.725	0.752	0.823
yahoo	0.732	0.742	0.758	0.780	0.738	0.670	0.582
zdf	0.929	0.929	0.745	0.847	0.772	0.578	0.514

the CCV does not cause a drop in the performance. We can deduce that ACCB – though also improving the F1 performance on the wiki package significantly – is not overfitted for wiki style documents, but actually performs better for some of the other packages as well.

The next observation concerns the performance of ACCB in comparison with DSC. While for the original, character based CCB and for the token based TCCB the performance does not show clear advantages or disadvantages, ACCB in general performs better than DSC. Looking at the total of our 14 evaluation scenarios, ACCB is scoring significantly higher F1 scores than DSC for five packages, comparable results on six packages and worse results only for three packages. Further, on four of the comparable packages ACCB is having slightly higher F1 scores, which might underline the tendency of a better performance.

The reason for this better performance is that ACCB obtains better recall values than DSC. So, even if it is in comparison slightly less precise, in the light of F1 it reaches a better tradeoff between recall and precision. However, the recall of ACCB is not perfect: if parts of the main content

Table 3. Average processing time in s/kB

	ACCB	CCB	TCCB	DSC	Crunch	LQF	Plain
bbc	0.001	0.007	0.001	0.001	0.027	0.004	0.000
chip	0.008	0.008	0.001	0.001	0.012	0.003	0.000
economist	0.015	0.015	0.002	0.002	0.014	0.016	0.000
espresso	0.016	0.016	0.002	0.002	0.033	0.010	0.000
golem	0.009	0.009	0.001	0.001	0.033	0.004	0.000
heise	0.012	0.011	0.001	0.001	0.032	0.011	0.000
manual	0.020	0.018	0.001	0.001	0.047	0.017	0.000
repubblica	0.014	0.011	0.001	0.001	0.048	0.001	0.000
slashdot	0.013	0.013	0.001	0.001	0.019	0.009	0.000
spiegel	0.015	0.016	0.001	0.001	0.018	0.009	0.000
telepolis	0.052	0.052	0.004	0.003	0.077	0.051	0.000
wiki	0.028	0.024	0.003	0.002	0.073	0.013	0.000
yahoo	0.013	0.013	0.001	0.001	0.024	0.011	0.000
zdf	0.001	0.001	0.001	0.001	0.028	0.001	0.000

are highly formatted, they might not be recognised. Especially if they are positioned close to other highly structured additional contents, the situation is very difficult for all content code blurring implementations. Typically, this problem occurs for headlines of news articles. They usually follow a section of additional contents, have a short text themselves and are separated from the rest of the main content by some other formatting instructions. However, DSC suffers from the same problem, even to a higher degree. ACCB's recall improvement is also due to a better performance exactly in these cases. Our flexible region approach seem to be more suitable for CE than the sliding windows of DSC.

On the slashdot package, instead, ACCB is inferior to DSC. Here, ACCB still achieves better recall values but is much less precise. The reason for this phenomenon are the short main content and the long additional texts of the user discussion in these documents. ACCB extracts the discussion text as well and better than DSC. But, as they are considered additional content according to the gold standard, ACCB is penalised in the precision rating.

Interesting is also that on the same data DSC comes second to TCCB when considering the F1 performance. This observation can be made also for the spiegel package, where ACCB is outperformed by DSC, but TCCB obtains still better results. So, the question is, if in these cases it is merely the character based approach with has exceptional drawbacks in comparison with a token based one. This might be a hint, that an intermediate approach for constructing the CCV could improve the results of our algorithm. A character based solution which restricts the impact of long words and tags is an idea to follow.

Looking finally at the processing time in table 3, the token based approaches of DSC and TCCB are comparably fast. ACCB and CCB need longer to process a document, simply due to the much longer CCV. However, they are both reasonably fast: their time performance is similar to LQF and still better than Crunch.

5. Conclusions and Future Work

We presented content code blurring, a CE algorithm which can be based either on characters or tokens. Comparing it with other CE algorithms we have shown that on most documents it yields better results. Especially on the difficult wiki style documents the adapted version ACCB is clearly superior to DSC. The results in absolute terms could still be improved, though.

Future works will comprise fine tuning of the parameters and to incorporate some notion of the DOM block elements to enhance recognition of structurally related parts of a document. Further, we will do some experiments with different models for constructing the CCV.

References

[1] S. Debnath, P. Mitra, and C. L. Giles. Identifying content blocks from web documents. In *Foundations of Intelligent Systems*, Lecture Notes in Computer Science, pages 285–293, 2005.

[2] A. Finn, N. Kushmerick, and B. Smyth. Fact or fiction: Content classification for digital libraries. In *DELOS Workshop: Personalisation and Recommender Systems in Digital Libraries*, 2001.

[3] D. Gibson, K. Punera, and A. Tomkins. The volume and evolution of web page templates. In *WWW '05: Special Interest Tracks and Posters of the 14th International Conference on World Wide Web*, pages 830–839, New York, NY, USA, 2005. ACM Press.

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

[5] S. Gupta, G. Kaiser, D. Neistadt, and P. Grimm. DOM-based content extraction of HTML documents. In *WWW '03: Proceedings of the 12th International Conference on World Wide Web*, pages 207–214, New York, NY, USA, 2003. ACM Press.

[6] C. Mantratzis, M. Orgun, and S. Cassidy. Separating XHTML content from navigation clutter using DOM-structure block analysis. In *HYPERTEXT '05: Proceedings of the sixteenth ACM conference on Hypertext and hypermedia*, pages 145–147, New York, NY, USA, 2005. ACM Press.

[7] D. Pinto, M. Branstein, R. Coleman, W. B. Croft, M. King, W. Li, and X. Wei. QuASM: a system for question answering using semi-structured data. In *JCDL '02: Proceedings of the 2nd ACM/IEEE-CS joint conference on Digital libraries*, pages 46–55, New York, NY, USA, 2002. ACM Press.

[8] A. F. R. Rahman, H. Alam, and R. Hartono. Content extraction from HTML documents. In *WDA 2001: Proceedings of the First International Workshop on Web Document Analysis*, pages 7–10, 2001.