

Combinations of Content Extraction Algorithms

Yves Weissig¹, Thomas Gottron²

¹Technische Universität Darmstadt, 64289 Darmstadt, Germany

²Johannes Gutenberg-Universität Mainz, 55099 Mainz, Germany
weissig@rbg.informatik.tu-darmstadt.de, gottron@uni-mainz.de

Abstract

Content Extraction is the task to identify the main text content in web documents – a topic of interest in the fields of information retrieval, web mining and content analysis. We implemented an application framework to combine different algorithms in order to improve the overall extraction performance. In this paper we present details of the framework and provide some first experimental results.

1 Introduction

HTML documents on the web are composed of far more data than their main content. Navigation menus, advertisements, functional or design elements are typical examples of additional contents which extend, enrich or simply come along with the main content. This “noise” in the documents contributes for about 40 to 50% of the data on the World Wide Web [Gibson *et al.*, 2005].

Cleaning web documents from this additional contents improves performance of information retrieval, web mining and content analysis applications. The task of identifying and/or extracting the main text content of a web document is most commonly referred to as Content Extraction (CE). Several good algorithms have been developed and evaluated in the last years. However, only the Crunch system [Gupta *et al.*, 2003] tried to approach the task with a fixed ensemble of different algorithms. As the heuristics employed in Crunch performed better in combination than when they are used individually, we motivated in [Gottron, 2008a] to look closer at the potential of using ensembles of CE algorithms. This led to the idea of the proposed CombinE system as outlined in figure 1. Its aim is to flexibly incorporate the results of filter modules implementing different algorithms and, thereby, obtain better or more reliable extracts of the main content.

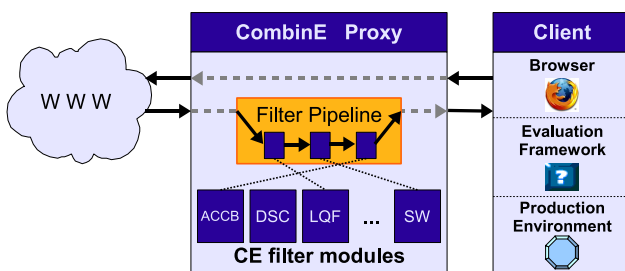


Figure 1: Outline of the *CombinE* system.

In this paper we show how combinations of CE algorithms can be realised and present some first results on how successful such combinations can be.

We proceed with a look at related work in the next section and a short description of our application in section 3. Then we focus on merging different document extracts in 4 and take a look at some first CE combinations and their performance in 5, before concluding the paper in 6 with a prospect at future work.

2 Related Work

The number of CE algorithms available for cleaning HTML documents is increasing. Especially in the last years the topic seems to receive more attention than before, probably due to the increased amount of noise in web documents.

The *Body Text Extraction* (BTE) algorithm [Finn *et al.*, 2001] interprets an HTML document as a sequence of word and tag tokens. It identifies a single, continuous region which contains most words while excluding most tags. A problem of BTE is its quadratic complexity and its restriction to discover only a single and continuous text passage as main content. Pinto *et al.* extended BTE in their *Document Slope Curves* (DSC) algorithm [Pinto *et al.*, 2002]. Using a windowing technique they are capable to locate also several document regions in which the word tokens are more frequent than tag tokens, while also reducing the complexity to linear runtime. *Link Quota Filters* (LQF) are a quite common heuristic for identifying link lists and navigation elements. The basic idea is to find DOM elements which consist mainly of text in hyperlink anchors. Mantratzis *et al.* presented a sophisticated LQF version in [Mantratzis *et al.*, 2005], while Prasad and Paepcke describe how to learn a weighting scheme for LQF in [Prasad and Paepcke, 2008]. *Content Code Blurring* (CCB) [Gottron, 2008b], instead, is based on finding regions in the source code character sequence which represent homogeneously formatted text. Its ACCB variation, which ignores format changes caused by hyperlinks, performed better than all previous CE heuristics. Weninger and Hsu proposed a line based approach to find text-rich areas in [Weninger and Hsu, 2008].

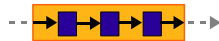
Evaluation of CE algorithms was addressed in [Gottron, 2007]. Providing a goldstandard for the main content, the idea is to compute the intersection of a CE algorithms’ output with the goldstandard using the longest common subsequence [Hirschberg, 1975] over the word sequences. This allows to apply classical IR measures like Recall, Precision and F_1 . The fact, that such an evaluation can be run automatically and unsupervised is exploited in [Gottron, 2009] to construct a framework for parameter optimisation based on genetic algorithms.

3 The CombinE System

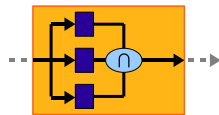
The aim to flexibly combine different CE algorithms in order to obtain better and/or more reliable results motivated the idea of the CombinE System [Gottron, 2008a]. It is an http-proxy server which can filter documents on-the-fly when receiving them from the web. An outline of the overall system was already given in figure 1.

To filter the web documents, CombinE uses an extensible set of CE algorithms with a standardised interface. The combination of algorithms is modelled in filter sets which can be realised in different ways:

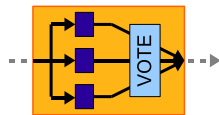
Serial The CE algorithms are applied in a predefined order, where the output document of one filter provides the input for the next one.



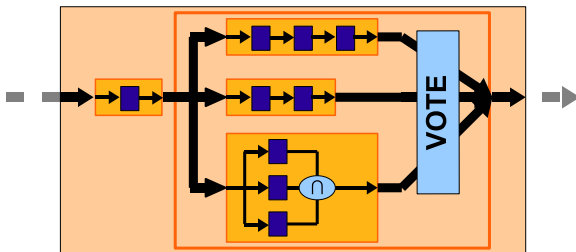
Parallel Several CE algorithms are applied to individual copies of the original document separately. The results are then combined by either unifying or intersecting the results. We get to the details of how to do this in the next section.



Voting A special case of the parallel combination is a voting setup. Each CE filter can vote for parts of a web document to belong to the main content. Only if a content fragment gets enough votes from the filters, it is finally declared to actually be main content. Attaching weights to the votes provides a further possibility to influence the results.



As technically each of this combinations of CE algorithms acts like a CE algorithm itself (HTML document as input and cleaned HTML as output), they can be incorporated in other filter pipelines. The following example demonstrates how filter sets can be combined into new, more complex sets.



The CombinE proxy can be configured comfortably over a web interface (c.f. figure 2). Users can choose from a selection of standalone CE algorithms as well as already combined versions to create new serial, parallel or voting filter sets.

4 Merging Extraction Results

The implementation of the different merging methods used within CombinE was one of the most challenging tasks in

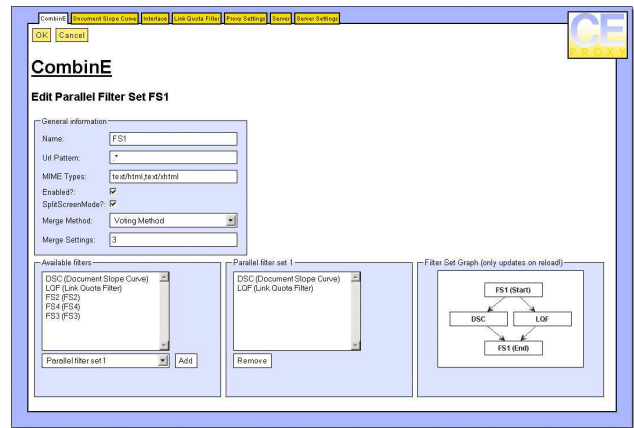


Figure 2: Web interface for creating and managing filter sets.

the project. While the serial approach was easy to implement because the output of the previous filter was used as the input of the next filter, the parallel combination is a non-trivial issue. A filter set can have up to N different sub filters which need to be executed. After each of the N filters returned a result it is the task of the merging method to integrate the N results into one message that is returned to the user.

CombinE uses a merging method based on DOM-trees. The generation of a DOM-tree out of an http-message is obtained via the NekoHTML parser¹, which is flexible, fast and reliable. We also use NekoHTML to normalise the HTML documents into a form which is compliant with W3C specifications.

The overall process of parallel filtering can be subdivided into the following steps:

- Pre-process the http-message, normalise HTML.
- Filter copies of the pre-processed http-message with each filter. The result is an array containing N filtered http-messages.
- Convert each of the filtered messages into DOM-trees.
- Merge the N DOM-trees with the merging method chosen for the current filter pipeline. The result is a single DOM-tree.
- Transform the DOM-tree into an http-message.
- Return the resulting http-message.

We consider the DOM-tree as a tree with M nodes. The nodes represent different HTML-elements, e.g. a `<table>`-Tag, an ``-Tag or a text-value. The intersection merging returns only elements appearing in each of the N filtered documents. The result can be described as $R_{\cap} := \{x \mid \forall i \in N : x \in D_i\}$ with x the nodes of the DOM-trees, so that R is the set of all elements contained in all D_i . In contrast to this, the union merging returns the elements that appear in any one of the filtered documents. It can be described as $R_{\cup} := \{x \mid \exists i \in N : x \in D_i\}$.

While the calculation of general tree mapping is a complex task, we can assume the CE filters to at most remove nodes from the DOM-tree. Hence, in an recursive approach, we start from the DOM-tree root nodes and collect all child nodes that are present in all (intersection) or any (union) of the DOM-trees.

¹<http://sourceforge.net/projects/nekohtml>

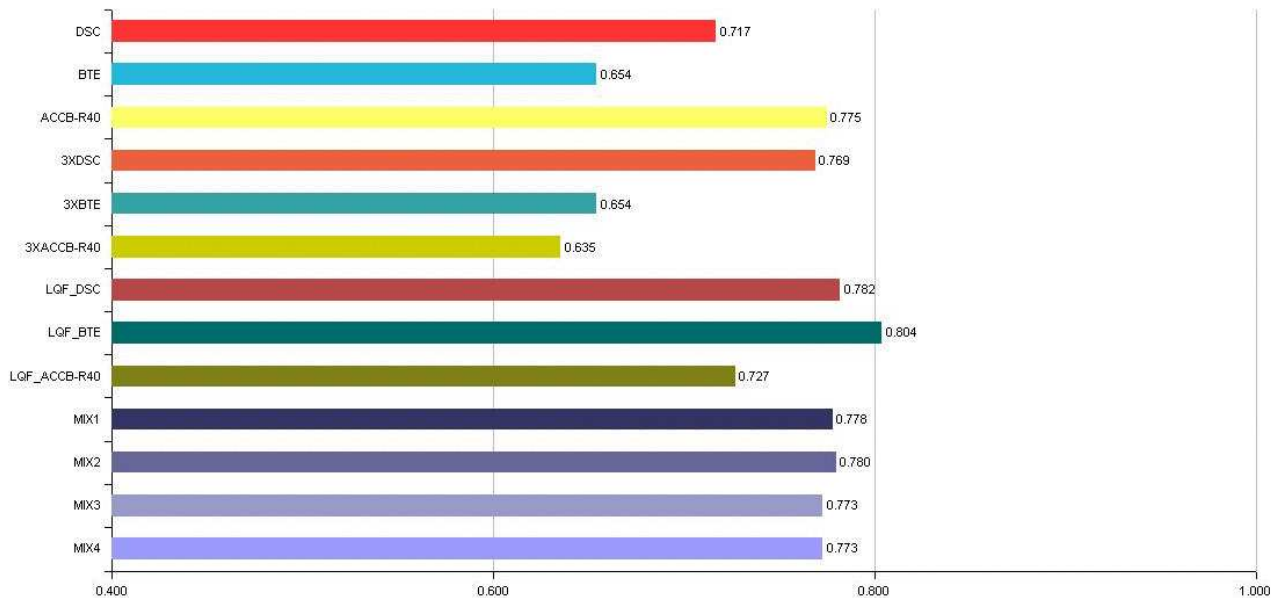


Figure 3: Diagram showing the average F1 results of the evaluation.

In a voting setup we additionally calculate the received (potentially weighted) votes for each DOM node and check whether they reach a value above the predefined threshold for inclusion in the resulting document.

5 Some first results

To evaluate the effectiveness of CE algorithm combinations, we use an existing corpus of 9.601 HTML documents. For each of those documents the main content is provided as gold standard. To compare the overlap between an extract and the gold standard we calculate the longest common (not necessarily continuous) subsequence of words in both texts. Using the number of words in the computed extract e , in the gold standard g and in their overlap $e \cap g = lcss(e, g)$, we can adopt the classical IR measures recall (r), precision (p) and the F1 measure (f_1) for CE evaluation:

$$r = \frac{|e \cap g|}{|g|}, p = \frac{|e \cap g|}{|e|}, f_1 = \frac{2 \cdot p \cdot r}{r + p}$$

In previous experiments we discovered the DSC, BTE, ACCB and LQF approaches to provide good results. Hence, we focused on combinations of those algorithms. Further, as LQF follows a rather different approach to CE, we constructed serial pipelines of LQF and the other three algorithms. This left us with several filter pipelines to evaluate:

3XDSC Serial execution of three DSC instances.

3XBTE Serial execution of three BTE instances.

3XACCB-R40 Serial execution of three ACCB instances.

LQF_DSC Serial setup of LQF with a downstream DSC filter.

LQF_BTE Serial setup of LQF with a downstream BTE filter.

LQF_ACCB-R40 Serial setup of LQF with a downstream ACCB filter.

MIX1 Parallel setup of DSC, BTE and ACCB with union merging of the results.

MIX2 Parallel setup of DSC, BTE and ACCB with intersection merging of the results.

MIX3 Parallel setup of DSC, BTE and ACCB with voted merging of the results, where at least two of the algorithms had to vote for an element to be included in the final document.

MIX4 Parallel setup of DSC, BTE and ACCB with weighted voted merging. DSC and BTE had a weight of 1, ACCB had a weight of 2 and the acceptance threshold was set to 2.

The detailed results of our experiments concerning the F_1 measure can be seen in table 1, a graphical representation of the average performance is shown in figure 3. Additionally to the above mentioned filter pipelines we included the performance of single BTE, DSC and ACCB filters for reference with previous results.

First of all we can observe that several combinations yield equivalent or slightly better results than a filter based on a single ACCB instance, which so far was the best general purpose CE-filter. The serial combination of LQF and BTE even delivers significantly better extracts. An explanation might be, that LQF removes navigation menus and links lists that are embedded into the main content. Hence, BTE can afterwards detect a continuous region of main content more easily. In this combination also the issue of BTE's quadratic runtime complexity is less pressing. As LQF already removes a good portion of navigation menus and alike, BTE operates on a much smaller document. Therefore, on most real-world documents (size between 60 and 100Kb) this combination takes between 1 and 1,5 seconds for extracting the main content – an acceptable performance in most application scenarios.

The parallel filters in MIX1 through MIX4 did not provide significant improvements. However, their performance is more stable across all evaluation packages and, thus, across different document styles. The triple serial execution of BTE, DSC and ACCB fostered precision at the cost of recall. Concerning F_1 , this lead to improvements only for 3XDSC.

Table 1: Word Sequence F1 Results

	bbc	chip	economist	espresso	golem	heise	manual	repubblica	slashdot	spiegel	telepolis	wiki	yahoo	zdf
ACCB-R40	0.718	0.703	0.890	0.875	0.959	0.916	0.423	0.968	0.177	0.861	0.908	0.682	0.847	0.929
BTE	0.676	0.262	0.736	0.835	0.532	0.674	0.409	0.842	0.113	0.753	0.927	0.856	0.663	0.875
DSC	0.595	0.173	0.881	0.862	0.958	0.877	0.403	0.925	0.252	0.902	0.859	0.594	0.906	0.847
3XACCB-R40	0.718	0.261	0.792	0.798	0.568	0.686	0.425	0.642	0.132	0.726	0.880	0.775	0.727	0.764
3XBTE	0.676	0.262	0.736	0.835	0.532	0.674	0.409	0.842	0.113	0.752	0.927	0.854	0.663	0.875
3XDSC	0.931	0.716	0.863	0.867	0.949	0.870	0.392	0.914	0.258	0.900	0.795	0.572	0.901	0.832
LQF_ACCB-R40	0.860	0.591	0.857	0.805	0.910	0.870	0.413	0.680	0.144	0.843	0.911	0.663	0.821	0.809
LQF_BTE	0.972	0.846	0.884	0.871	0.988	0.880	0.396	0.972	0.143	0.834	0.941	0.710	0.898	0.926
LQF_DSC	0.908	0.709	0.881	0.873	0.984	0.898	0.406	0.945	0.239	0.888	0.864	0.561	0.908	0.882
MIX1	0.938	0.822	0.871	0.849	0.942	0.869	0.401	0.895	0.252	0.883	0.856	0.605	0.903	0.803
MIX2	0.936	0.824	0.872	0.866	0.954	0.868	0.380	0.915	0.252	0.893	0.854	0.596	0.891	0.823
MIX3	0.938	0.822	0.871	0.849	0.953	0.867	0.406	0.895	0.252	0.895	0.856	0.605	0.902	0.717
MIX4	0.938	0.822	0.871	0.849	0.953	0.867	0.406	0.895	0.252	0.895	0.856	0.605	0.902	0.717

6 Conclusions and Future Work

The CombinE System is capable to combine content extraction algorithms in different ways and setups. It is designed as an http-proxy which allows an easy and transparent incorporation of content filtering in IR systems accessing the web. We found some first interesting results, where filter combinations achieved better results than single filters.

A topic of ongoing and future research is the discovery and optimisation of good filter pipelines. The genetic algorithm framework for parameter optimisation [Gottron, 2009] is currently extended to explore also filter combinations. The aim is to automatically tune and further improve content extraction on HTML documents.

References

- [Finn *et al.*, 2001] Aidan Finn, Nicholas Kushmerick, and Barry Smyth. Fact or fiction: Content classification for digital libraries. In *DELOS Workshop: Personalisation and Recommender Systems in Digital Libraries*, 2001.
- [Gibson *et al.*, 2005] David Gibson, Kunal Punera, and Andrew 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.
- [Gottron, 2007] Thomas Gottron. Evaluating content extraction on HTML documents. In *ITA '07: Proceedings of the 2nd International Conference on Internet Technologies and Applications*, pages 123–132, September 2007.
- [Gottron, 2008a] Thomas Gottron. Combining content extraction heuristics: the combine system. In *iiWAS '08: Proceedings of the 10th International Conference on Information Integration and Web-based Applications & Services*, pages 591–595, New York, NY, USA, 2008. ACM.
- [Gottron, 2008b] Thomas Gottron. Content code blurring: A new approach to content extraction. In *DEXA '08: 19th International Workshop on Database and Expert Systems Applications*, pages 29 – 33. IEEE Computer Society, September 2008.
- [Gottron, 2009] Thomas Gottron. An evolutionary approach to automatically optimise web content extraction. In *IIS'09: Proceedings of the 17th International Conference Intelligent Information Systems*, pages 331–343, 2009.
- [Gupta *et al.*, 2003] Suhit Gupta, Gail Kaiser, David Neistadt, and Peter 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.
- [Hirschberg, 1975] D. S. Hirschberg. A linear space algorithm for computing maximal common subsequences. *Commun. ACM*, 18(6):341–343, 1975.
- [Mantratzis *et al.*, 2005] Constantine Mantratzis, Mehmet Orgun, and Steve 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.
- [Pinto *et al.*, 2002] David Pinto, Michael Branstein, Ryan Coleman, W. Bruce Croft, Matthew King, Wei Li, and Xing 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.
- [Prasad and Paepcke, 2008] Jyotika Prasad and Andreas Paepcke. CoreEx: content extraction from online news articles. In *CIKM '08: Proceeding of the 17th ACM conference on Information and knowledge management*, pages 1391–1392, New York, NY, USA, 2008. ACM.
- [Weninger and Hsu, 2008] Tim Weninger and William H. Hsu. Text extraction from the web via text-tag-ratio. In *TIR '08: Proceedings of the 5th International Workshop on Text Information Retrieval*, pages 23 – 28. IEEE Computer Society, September 2008.