Combining Content Extraction Heuristics: The *CombinE* System

Thomas Gottron
Institut für Informatik
Johannes Gutenberg-Universität Mainz
55099 Mainz, Germany
gottron@uni-mainz.de

ABSTRACT

The main text content of an HTML document on the WWW is typically surrounded by additional contents, such as navigation menus, advertisements, link lists or design elements. Content Extraction (CE) is the task to identify and extract the main content. Ongoing research has spawned several CE heuristics of different quality. However, so far only the Crunch framework combines several heuristics to improve its overall CE performance. Since Crunch, though, many new algorithms have been formulated. The *CombinE* system is designed to test, evaluate and optimise combinations of CE heuristics. Its aim is to develop CE systems which yield better and more reliable extracts of the main content of a web document.

Categories and Subject Descriptors

H.3 [**Information Storage and Retrieval**]: Information Search and Retrieval, Systems and Software

Keywords

Content Extraction, filter ensembles, evaluation

1. INTRODUCTION

Modern web documents contain 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. Figure 1 shows a representative web document in which the main content has been outlined to distinguish it from the additional contents. This example also illustrates that the additional contents contribute a lot to the overall content of a web document. Gibson et al. [6] estimated them to account for 40 to 50% of the data on the WWW, predicting this ratio to increase constantly.

While for human users these additional contents do not pose a problem in the perception of web documents, they are difficult to handle when accessing the information of a document automatically. Web Mining [1], web based Information Retrieval [22], Natural Language Processing [4], enhancing access for screen readers or handheld devices [12], information extraction [15] and content integration [23] are some scenarios where the main content needs to be separated from the additional contents in an automatic way.

Figure 1: An example for a web document with an outlined main content.

The process of determining the main content of an HTML document is commonly referred to as *Content Extraction* (CE). In the last years, several CE heuristics have been formulated. Some of the existing and three new approaches will be mentioned and explained briefly in this paper. Most CE approaches, however, stand alone and isolated. An attempt to combine different heuristics into one system was made only by the Crunch framework [13]. Crunch demonstrated that a well chosen combination of different CE algorithms can provide better results than any of the single approaches on its own. But, since Crunch several new CE algorithms have been developed, while others have been refined. Also concerning the evaluation of CE systems some new results

iiWAS2008 , November 24–26, 2008, Linz, Austria.

have been published [7]. Given these advances in the field, it is necessary to analyse the potential of combining CE heuristics in a more systematic way.

In this paper we will describe the *CombinE* system, which allows to easily configure ensembles of CE algorithms and test their combined performance based on objective criteria. As the evaluation tests can be run automatically and provide quantitative results, the system also includes an option to optimise the configuration of the CE algorithms.

We proceed as follows: in section 2 we give an overview of related work, followed with the brief description of some new CE heuristics in 3. We continue with an outline of the *CombinE* system in 4, where we will go into the details of how it works and which combinations of CE heuristics it allows. We also describe, which parts of the system do already exist and which parts are still worked on, before summarising the ideas in 5.

2. RELATED WORK

The expression Content Extraction was introduced by Rahman et al. in [21]. Aside from an outline for a CE algorithm, the authors contributed the first definition of the task and some generic quality criteria for CE systems.

Finn et al. presented the *Body Text Extraction* (BTE) algorithm in [5]. They interpret an HTML document as a sequence of word and tag tokens and identify a single, continuous region which contains most words while excluding most tags. Pinto et al. [20] extended this approach to construct *Document Slope Curves* (DSC). Using a windowing technique, they locate several document regions in which the word tokens are more frequent than tag tokens. Though DSC was originally not designed for CE, we showed its applicability for extracting the main content in [7].

Link Quota Filters (LQF) are a common approach to identify link lists and navigation elements. They identify DOM elements which have a high ratio of text residing in hyperlink anchors. Mantratzis et al. presented a sophisticated LQF version in [19].

Han et al. developed the *Largest Size Increase* (LSI[1]) algorithm for their wrapper generation system [15]. Being one of several techniques to find highly structured and data rich regions in a web document, LSI identifies the DOM subtree which contributes strongest to the visible content in an HTML document. Hence, it is suitable for CE tasks, too.

Also the above mentioned *Crunch* framework [12, 14, 13, 11] incorporates an LQF implementation. However, Crunch avoids a one-solution-fits-all approach and combines several heuristics to filter documents, e.g. by removing link and text lists, advertisements, etc. Crunch's main objective is to optimise HTML documents for displaying them on handheld devices or to improve accessibility for screen readers.

Debnath et al. developed *FeatureExtractor* (FE) and *K-FeatureExtractor* (KFE) in [2, 3] and based them on a segmentation of the document into blocks. Each block is analysed for particular features like the amount of text, the presence of images, script code, etc. The extraction process is a selection of those blocks which correspond best to a desired feature, e.g. the presence of long texts.

The *Content Code Blurring* (CCB) algorithm was introduced in [10]. It is based on finding regions in the source code character sequence which represent homogeneously formatted text. In its ACCB version, which ignores format changes caused by hyperlinks, it performs better than all other CE heuristics. A similar approach was analysed by Weninger and Hsu in [24]. They consider the source code and calculate the line based *text-to-tag ratio (TTR)*. Smoothing the TTR across several source code lines, they find content rich parts of web documents.

A different approach to CE are *Template Detection* (TD) algorithms [18, 25, 17, 22, 1]. They use collections of training documents based on the same template to identify a common structure and extract the main content via removing the identical parts found in all documents. By applying template based clustering of web documents [9] we showed in [8] that it is possible to convert any TD algorithm into a single document CE algorithm by building suitable training sets automatically. However, so far this process is too time consuming to incorporate it in online extraction systems, where the results have to be provided without long delays.

3. NEW CONTENT EXTRACTION HEURISTICS

After having mentioned the most prominent existing CE algorithms in the last section, we will now describe a few algorithms which are genuine or have been motivated from other fields of research.

Largest Pagelet (LP) Bar-Yossef and Rajagopalan [1] defined *pagelets* as self-contained regions in a web document. This semantic definition suggests the main content to be a pagelet – most likely the largest in a document. Hence, LP uses the *PagePartitioning* algorithm to locate the pagelets in a web document and selects the one containing the most text. This largest pagelet is then extracted as the main content of the document.

Document Tree Plateus (DTP) Combining the idea of DSC with a DOM tree structure analysis leads to the DTP approach. Plotting the DOM tree depth against the DSC token sequence in a document allows to detect plateaus of content rich regions in a document. Using the additional knowledge of the tree structure, DTP finds the main content by looking for plateaus in different branches of the same DOM sub-tree.

Stop Word (SW) ratio This CE algorithm is based on the observation that the main content usually is a well formulated text. Additional contents, instead, consist of loosely connected keywords or short texts. Hence, the main content typically shows a relatively high rate of stop words. Thus, SW determines the stop word ratio in each text node of the DOM tree and deletes all those nodes with a ratio below a given threshold.

4. THE *COMBINE* SYSTEM

In [7] we showed that Crunch's combination of LQF with other lightweight CE approaches generally improves the detection of the main content. The aim of our *CombinE* system is to explore systematically the potential of combining CE heuristics. In particular, it allows to configure and evaluate such combinations in an easy, even automated way.

[1]Note that the largest size increase algorithm is entirely different from the concept of latent semantic indexing which is commonly abbreviated by LSI as well.

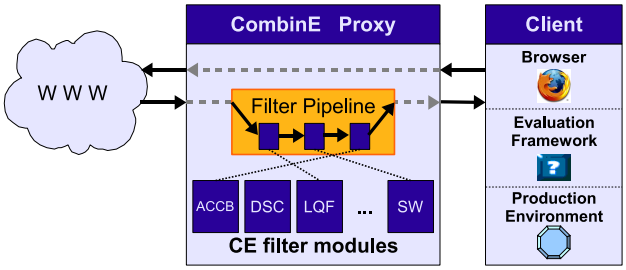

Figure 2: Outline of the *CombinE* system.

The outline of the system is shown in figure 2. Like Crunch, also *CombinE* will be set up as an HTTP proxy server. This has the advantage, that it can be used as a local proxy to transparently extract the main content from web documents while browsing the web. Hence, the performance of a CE filter combination can easily be evaluated manually by simply looking at filtered web documents in a browser. Thanks to the wide spread HTTP protocol, the proxy structure also allows an easy integration with other applications. Therefore, *CombinE* can be used as production system as well, once a suitable combination of filters has been developed.

Internally, *CombinE* consists mainly of an extensible collection of *CE filter modules* with a standardised interface. Those modules are the building blocks for the eventual *filter pipelines*.

4.1 Filter pipelines

A filter pipeline allows to combine an arbitrary number of CE filter modules. However, it must contain at least one of them. The filter modules can be combined in different ways, depending on the type of the pipeline:

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

Parallel Several CE filters are applied to individual copies of the original document separately. The results are then combined by either unifying or intersecting the results, i.e. by extracting only those texts which were extracted by any one or by all filters.

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

As technically, each pipeline constitutes a CE filter itself, they can be reused in other pipelines. It is, for instance, possible to have the results of serial and parallel pipelines combined by a voting mechanism.

4.2 Evaluation and Optimisation

The proxy structure of *CombinE* also allows to employ an existing evaluation framework [7]. This framework can be used just the same to evaluate the CE performance of an entire filter pipeline instead of a single CE algorithm.

Part of the evaluation framework is a growing collection of document packages for different scenarios, i.e. web document styles. Each package contains several documents, usually based on the same template. Except the original HTML documents the packages also provide a gold standard for their main content. Table 1 gives an overview of the packages which have been compiled so far. Their documents were chosen carefully to cover a wide variety of web page layouts, main content lengths and languages.

Table 1: Evaluation packages

Package	URL	Documents
bbc	http://news.bbc.co.uk	1,000
chip	http://www.chip.de	361
economist	http://www.economist.com	250
espresso	http://espresso.repubblica.it	139
golem	http://golem.de	1,000
heise	http://www.heise.de	1,000
manual	several	65
repubblica	http://www.repubblica.it	1,000
slashdot	http://slashdot.org	364
spiegel	http://www.spiegel.de	1,000
telepolis	http://www.telepolis.de	1,000
wiki	http://de.wikipedia.org	1,000
yahoo	http://news.yahoo.com	1,000
zdf	http://www.heute.de	422
Total		*9,601*

To evaluate a CE filter – or a filter pipeline – the framework launches a small webserver to serve exactly one document. Then it retrieves this document via a CE proxy server which is responsible for extracting the main content of this document on-the-fly. Figure 3 visualises this process. The extract created in this way is compared to the gold standard by computing the longest common subsequence (LCS) [16] of words in both texts. This LCS defines an overlap between gold standard and extract, thereby allowing to compute classical Information Retrieval evaluation measures like recall, precision and F1. In this way, the performance of several single CE heuristics has already been evaluated. The results concerning the most important indicator, i.e. the F1 measure, are given in table 2.

Beyond its contribution of comparing the performance of filter pipelines, the evaluation framework can also be incorprated in a different role. By integrating it into the *CombinE* system, the feedback of its automatic evaluation procedure can be used to optimise the filter pipelines themselves.

One option for optimisation is to adjust the parameters of single CE modules and evaluate immediately the effects on

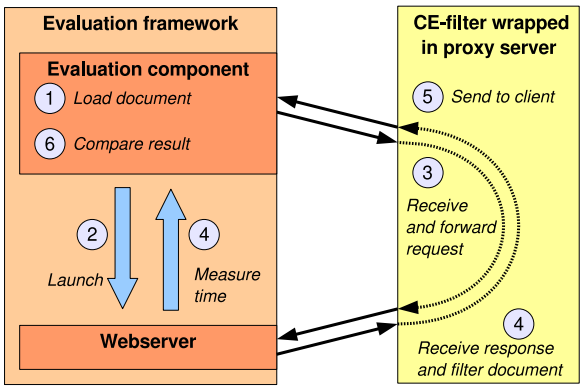

Figure 3: Evaluation process involving a CE proxy.

Table 2: Evaluation of single CE modules: average F1

	bbc	chip	economist	espresso	golem	heise	manual	republica	slashdot	spiegel	telepolis	wiki	yahoo	zdf	*Average*
ACCB	0.924	0.703	0.890	0.875	0.959	0.916	0.419	0.968	0.177	0.861	0.908	0.682	0.732	0.929	*0.782*
BTE	0.676	0.262	0.736	0.835	0.532	0.674	0.409	0.842	0.113	0.749	0.927	0.853	0.602	0.875	*0.649*
CCB	0.923	0.716	0.914	0.876	0.939	0.841	0.420	0.964	0.160	0.858	0.913	0.403	0.742	0.929	*0.757*
Crunch	0.756	0.342	0.815	0.810	0.837	0.810	0.382	0.887	0.123	0.706	0.910	0.725	0.738	0.772	*0.687*
DSC	0.937	0.708	0.881	0.862	0.958	0.877	0.403	0.925	0.252	0.902	0.859	0.594	0.780	0.847	*0.770*
DTP	0.706	0.711	0.740	0.759	0.877	0.791	0.389	0.915	0.149	0.767	0.942	0.635	0.810	0.739	*0.709*
FE	0.147	0.015	0.002	0.035	0.273	0.264	0.141	0.099	0.067	0.002	0.143	0.236	0.109	0.015	*0.111*
K-FE	0.677	0.276	0.697	0.035	0.200	0.580	0.357	0.097	0.077	0.689	0.823	0.593	0.673	0.491	*0.448*
LP	0.954	0.698	0.835	0.863	0.319	0.603	0.322	0.918	0.177	0.893	0.931	0.574	0.060	0.623	*0.626*
LQF	0.834	0.502	0.732	0.667	0.926	0.791	0.387	0.826	0.135	0.790	0.906	0.690	0.708	0.579	*0.677*
LSI	0.886	0.070	0.842	0.838	0.554	0.645	0.376	0.877	0.076	0.722	0.942	0.764	0.781	0.574	*0.639*
SW	0.741	0.544	0.776	0.875	0.770	0.830	0.409	0.835	0.126	0.756	0.926	0.540	0.793	0.848	*0.698*
TCCB	0.914	0.842	0.903	0.871	0.947	0.821	0.404	0.918	0.269	0.910	0.902	0.660	0.758	0.745	*0.776*

the extraction performance. Another option is to iterate automatically through different combinations of filter modules and see which one performs best.

Unfortunately, the exploration of the parameter space of even a single CE filter is a time consuming task. Likewise the number of possible combinations of CE filters in a pipeline is very large. Genetic programming could be a suitable solution to this problem, as the extraction performance is a self-evident measure for the fitness of a filter pipeline setup.

4.3 Current State of the System

Though, the *CombinE* system itself is under development, several of its components and some of its infrastructure are already in a ready-to-use state.

First of all, several CE filters have already been implemented and tested. Most of them have a standardised interface as they are used as content filters in a lightweight HTTP proxy. Also this proxy framework is reused in *CombinE*.

As shown above, the collection of evaluation data consists of 14 different packages with a total of 9,601 documents. While this test data will be extended and updated in the future, it provides already a good starting point.

Finally, the entire structure of the evaluation framework is present and can easily be incorporated into *CombinE*.

Though, these components are implemented, *CombinE* still lacks some "glue" between them to combine evaluation framework, proxy structure and CE filter modules.

A bit more work is needed for the filter pipelines. Though pretty straight forward in their description, the parallel approach to combine CE filters and moreover their output requires some efficient HTML document aligning techniques. A central part of *CombinE* will also be a suitable user interface for building or restructuring the pipelines.

Another open issue are the search strategies for finding optimal configurations of filter module parameters or pipeline combinations. Given the time consuming tasks of evaluating the extraction results of massive amounts of documents and CE filter combinations, a parallel or even distributed approach using the possibilities of multi-core processors or cluster architectures needs to be taken into consideration.

Finally, the collection of CE heuristics – and, thereby, filter modules – will have to be extended. The ongoing research in the field of CE yields always new ideas on how to detect the main content in a web document. Even if these heuristics might not perform outstanding on their own, they might still improve the extraction performance in combination with other CE filters.

5. SUMMARY AND OUTLOOK

This paper presented in detail the concept of the *CombinE* system, which is intended to analyse and evaluate the performance of combinations of CE heuristics. The aim is to construct ensembles of CE filters which at large perform better and more reliable than the single parts. Improving the performance of a CE system in general is of importance to all applications which analyse or integrate contents from web documents.

Several parts of the system do already exist, mainly a respectable collection of single CE filters, evaluation documents, several parts of the proxy infrastructure and the evaluation framework. The integration of those parts should not pose a big problem, thus, leaving the realisation of the CE filter pipelines and the genetic programming for the optimisation problems as major remaining tasks to complete the *CombinE* system.

6. REFERENCES

[1] Z. Bar-Yossef and S. Rajagopalan. Template detection via data mining and its applications. In *WWW '02: Proceedings of the 11th International Conference on World Wide Web*, pages 580–591, New York, NY, USA, 2002. ACM Press.

[2] S. Debnath, P. Mitra, and C. L. Giles. Automatic extraction of informative blocks from webpages. In *SAC '05: Proceedings of the 2005 ACM Symposium on Applied Computing*, pages 1722–1726, New York, NY, USA, 2005. ACM Press.

[3] 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.

[4] C. Fairon, H. Naets, A. Kilgarriff, and G.-M. de Schryver, editors. *WAC3: Proceedings of the 3rd web as corpus workshop, incorporating cleaneval*. Presses universitaires de Louvain, Sept. 2007.

[5] 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.

[6] 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.

[7] 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.

[8] T. Gottron. Bridging the gap: From multi document template detection to single document content extraction. In *EuroIMSA '08: Proceedings of the IASTED Conference on Internet and Multimedia Systems and Applications 2008*, pages 66–71. ACTA Press, Calgary, Mar. 2008.

[9] T. Gottron. Clustering template based web documents. In *ECIR '08: Proceedings of the 30th European Conference on Information Retrieval*, pages 40–51. Springer, Mar. 2008.

[10] T. Gottron. Content code blurring: A new approach to content extraction. In *TIR '08: Proceedings of the 5th International Workshop on Text Information Retrieval*, pages 29 – 33. IEEE Computer Society, Sept. 2008.

[11] S. Gupta, H. Becker, G. Kaiser, and S. Stolfo. Verifying genre-based clustering approach to content extraction. In *WWW '06: Proceedings of the 15th International Conference on World Wide Web*, pages 875–876, New York, NY, USA, 2006. ACM Press.

[12] 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.

[13] S. Gupta, G. Kaiser, and S. Stolfo. Extracting context to improve accuracy for HTML content extraction. In *WWW '05: Special Interest Tracks and Posters of the 14th International conference on World Wide Web*, pages 1114–1115, New York, NY, USA, 2005. ACM Press.

[14] S. Gupta, G. E. Kaiser, P. Grimm, M. F. Chiang, and J. Starren. Automating Content Extraction of HTML Documents. *World Wide Web*, 8(2):179–224, 2005.

[15] W. Han, D. Buttler, and C. Pu. Wrapping Web Data into XML. *SIGMOD Rec.*, 30(3):33–38, 2001.

[16] D. S. Hirschberg. A linear space algorithm for computing maximal common subsequences. *Commun. ACM*, 18(6):341–343, 1975.

[17] H.-Y. Kao, S.-H. Lin, J.-M. Ho, and M.-S. Chen. Mining Web Informative Structures and Contents Based on Entropy Analysis. *IEEE Transactions on Knowledge and Data Engineering*, 16(1):41–55, 2004.

[18] S.-H. Lin and J.-M. Ho. Discovering informative content blocks from web documents. In *KDD '02: Proceedings of the eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 588–593, New York, NY, USA, 2002. ACM Press.

[19] 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.

[20] 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.

[21] 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.

[22] D. C. Reis, P. B. Golgher, A. S. da Silva, and A. F. Laender. Automatic web news extraction using tree edit distance. In *WWW '04: Proceedings of the 13th International Conference on World Wide Web*, pages 502–511, New York, NY, USA, 2004. ACM Press.

[23] F. Vitali, A. di Iorio, and E. V. Campori. Rule-based structural analysis of web pages. In *DAS 2004: Proceedings of the 6th International Workshop on Document Analysis Systems*, volume 3163 of *Lecture Notes in Computer Science*, pages 425–437. Springer, July 2004.

[24] T. Weninger and W. 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, Sept. 2008.

[25] L. Yi, B. Liu, and X. 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 Press.