

An Evolutionary Approach to Automatically Optimise Web Content Extraction

Thomas Gottron

Institut für Informatik, Johannes Gutenberg Universität Mainz, Germany

Abstract

Web content extraction is the task of identifying the main content of a web document. In the last years, research has spawned several algorithms to address this task and objective measures to evaluate the performance of such methods. The behaviour of many of these algorithms can be influenced via parameters, but time consuming evaluation has so far prevented a thorough manual or automatic finetuning and optimisation of the parameter settings. This paper presents an evolutionary approach as a suitable solution to find good parameter settings.

Keywords: content extraction, parameter optimisation, evolutionary computing

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. In figure 1 we have outlined the main content of a modern web document with a dashed line. Looking at this screenshot it becomes clear, how much data the additional contents contribute to the overall document.

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 or small screen devices.

In the last years, several algorithms for CE have been developed. Their performance has been tested and compared using objective criteria. Hence, users who are confronted with a CE problem can choose from a variety of methods of different quality and with different properties. The behaviour of most of the algorithms can be adapted via parameter settings. These parameter settings are usually set to default values provided by the authors who originally introduced the algorithms. But, this choice of the parameter settings is typically made following empirical observations.

Optimising the parameters of a CE algorithm in order to achieve the best performance is difficult. The evaluation of a CE algorithm involves extensive tests

The screenshot shows the BBC News website interface. At the top, there's a navigation bar with links for Home, News, Sport, Radio, TV, Weather, and Languages. Below that, there's a search bar and a 'Search' button. The main header area includes the BBC News logo, a 'One-Minute World News' button, and a 'News services' section. The article is titled 'New search powers lead Firefox 3' and is dated Tuesday, 26 February 2008. The article text is enclosed in a red dashed outline. The sidebar on the left lists various news categories like Africa, Americas, Asia-Pacific, Europe, Middle East, South Asia, UK, Business, Health, Science/Nature, Technology, Entertainment, and Also in the news. The right sidebar contains a 'SEE ALSO' section with links to related articles, a 'RELATED INTERNET LINKS' section, and a 'TOP TECHNOLOGY STORIES' section.

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

with several hundred or even thousand documents. A task which is very time consuming already for a single set of parameter values.

This problem is becoming even more pressing for CE systems like CombinE (Gottron, 2008a). CombinE allows to define combinations of CE algorithms with the aim to yield more reliable, more robust or even better results. Therefore, an exhaustive exploration of the parameter space becomes nigh impossible when including several algorithms with in turn comprise several numerical parameters. More even, the different possible ways of combining the algorithm themselves can be parameterised and, hence, optimised. Though, CombinE is still under development, the question of parameter optimisation is central to the system.

In this paper we propose an evolutionary computing approach as a solution to this task. Different parameter settings of a single algorithm provide the individuals of our population. Using well established evaluation measures from the field of IR to judge the performance of an algorithm configuration, we have a natural

definition of the fitness for these individuals. Recombining the settings of the fittest individuals of our population of parameter settings, we try to improve the performance and find a good if not even optimal solution. Operating on single algorithms is our test scenario for the genetic optimisation process, which we want to employ in the future for tuning CE algorithm combinations.

We proceed with a look at related work in the next section. Giving an overview of several CE algorithms we refer to the original papers for detailed information on how the algorithms work. In section 3 we briefly deal with the evaluation of CE systems and in 4 we describe our evolutionary framework. Section 5 describes some CE algorithms we used to test our framework and contains the results of these experiments. In 6 we conclude the paper in with a summary and a look at future work.

2 Related Work

A common approach to detect additional contents in the style of link lists and navigation menus are link quota filters (LQF), as described e.g. by Mantratzis *et al.* (2005). LQF use a high ratio of link content in document blocks to detect navigation menus and similar structures. Debnath *et al.* introduced the Feature-Extractor algorithm and its extension the K-FeatureExtractor in (Debnath *et al.*, 2005). 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. Weninger and Hsu (2008) use a line based text to tag ratio to identify text rich areas in web documents. Finn *et al.* (2001) described the Body Text Extraction (BTE) algorithm 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.* (2002) 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. To overcome the limitations of rigid windows we used an iterative approach of modelling the content-code ratio by calculating weighted local averages in (Gottron, 2008b). This content code blurring algorithm in its ACCB variation has proved to be the best performing CE algorithm to date.

Another prominent solutions for CE is the Crunch framework of Gupta *et al.* (2003). 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. The idea of obtaining better results with a combination of different CE algorithms motivated the already mentioned CombinE system (Gottron, 2008a). Except the serial application of a fixed set of filters, CombinE provides flexible content extraction pipelines with parallel, serial and voting mechanisms.

Rahman *et al.* (2001) were the first to 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 men-

tioned. In (Gottron, 2007) we developed a way to measure, evaluate and compare CE algorithms based on the common and objective IR measures precision, recall and F1.

Evolutionary programming and genetic algorithms come from the field of artificial intelligence and are a common method to solve difficult to handle optimisation problems. The book of Michalewicz (1996) provides a good introduction to the topic.

3 Evaluation of Content Extraction Algorithms

As our aim is to find parameter values for CE algorithms in order to obtain best extraction results, we need to describe first how to actually measure and evaluate the extraction performance. The overall approach, its motivation and the advantages compared to other evaluation methods are thoroughly discussed in (Gottron, 2007), so here we restrict ourselves to the essentials.

The entire evaluation is based on a set of test documents, for which we provide a gold standard for the actual main content. The test documents are fed into the CE algorithms and their output is compared to the gold standard. The comparison can essentially be based on the texts in the computed extract and the gold standard.

For each single test document, its text is represented as a sequence of words and the overlap between computed and actual main content is defined to be the longest common (not necessarily continuous) subsequence of words (Hirschberg, 1975). Using the number of words in the computed extract e , in the gold standard g and in their overlap $e \cap g = lc_{ss}(e, g)$, we can define the common IR measures recall (r), precision (p) and the F1 measure (f_1) for CE evaluation:

$$\begin{aligned} r &= \frac{|e \cap g|}{|g|} \\ p &= \frac{|e \cap g|}{|e|} \\ f_1 &= \frac{2 \cdot p \cdot r}{r + p} \end{aligned}$$

Example: We have got a document which in its full version contains the text *Title Copyright Some text in the body Commercial*. The gold standard defines the main content as *Title Some text in the body* and a CE algorithm provides *Title Copyright Some text in* as extract. The longest common subsequence is *Title Some text in*. Counting the words, this evaluates to a recall of $r = \frac{4}{6}$, a precision of $p = \frac{4}{5}$ and, hence, an F1 value of $f_1 = \frac{8}{11}$.

The biggest advantage of this evaluation approach – beside its objective measures – is that it can be completely automated. Once the gold standard is defined, the evaluation process can be run without any human interaction. This is a key feature for the optimisation via an evolutionary approach.

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

4 The Evolutionary Computing Framework

Given our initial remarks about the application background and the evaluation method explained above, an evolutionary approach for optimising CE algorithms can be set up rather straight forward. We provide a population of parameter settings for the algorithm and a set of test documents for evaluation. As fitness function to rate which parameter settings perform better than others, we use the average F1 value of the evaluation measures. The range of the F1 measure is between 0 and 1, where a value of 1 corresponds to a perfect extraction of the main content. Hence, it renders an ideal fitness function.

We implemented this process in a prototype application to test our ideas and to prove that it is feasible to optimise CE algorithms in this way. The prototype is extensible to different types and new kinds of parameters. In this way it will be possible in the future to optimise also combinations of CE filters, especially the different possible ways they can be combined with each other. The necessary user interaction with the application is kept at a minimum, to allow a very high degree of automation in the entire optimisation process.

In previous work (Gottron, 2007, 2008b) we created a large collection of 9.601 test documents. As shown in table 1, the documents are organised in different packages to reflect different scenarios and content languages. Instead of using such a large and complex collection of test documents, we restrict the evaluation for determining the fitness of individuals on a single, much smaller and more diversified set of documents. In this way we can evaluate each single parameter configuration relatively fast.

As the set of test documents can easily be replaced, it also allows a user to define own documents. In this way a user can optimise the algorithms for the very document scenario he is faced with. For this paper, instead, the small evaluation set is constructed by picking a representative random sample of documents from the different large evaluation packages. We can, however, use the large collection

to verify our findings of new and presumable better parameter settings.

To optimise a particular CE algorithm using evolutionary computing, we first need to describe the individuals. For this purpose, the user provides a description of the parameters which influence the algorithm. Such a parameter description lists all the parameters, contains information about their type and range, as well as a notion of how to setup the algorithm to use this parameters. Figure 2 outlines a parameter description, which can be interpreted as a blueprint for all individuals.

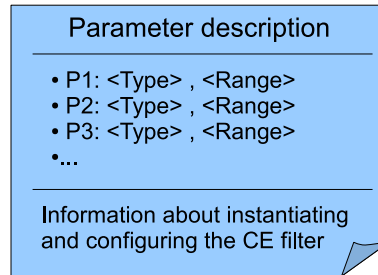


FIGURE 2: The parameter description provides information about the number, type and range of the parameters, as well as information about how to instantiate and configure the CE filter.

The optimisation process is shown in figure 3. It starts with a small population of parameter settings, which are randomly generated with respect to the range restrictions. Optionally, the user may add some individuals with manually defined parameter values, e.g. to inject individuals known to perform good. All individuals from this population are tested for their fitness, i.e. their settings are used to setup an instance of the CE algorithm, use this instance to filter a small collection of test documents and evaluate its average F1 performance.

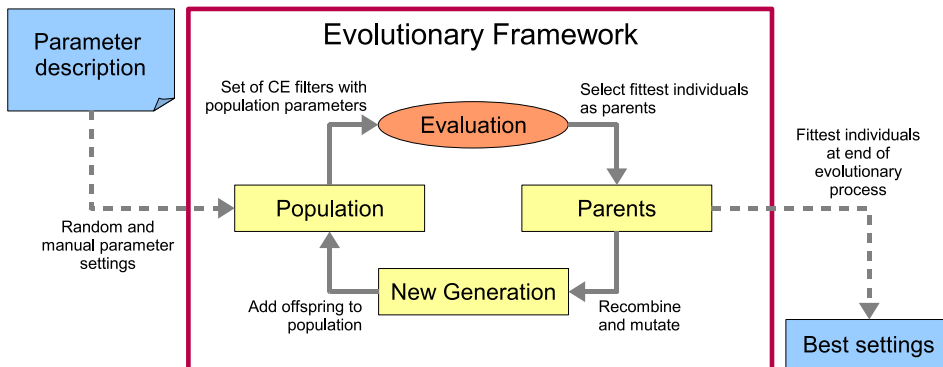


FIGURE 3: The framework of the evolutionary process.

In a next step, the better performing individuals are used as parents for the next generation of parameter settings. We allow the recombination of parameter values as well as mutation. So, we create new parameter settings, either be randomly

combining the values of two of the better individuals, or by mutating the values of such an individual. A mutation is realised by introducing random deviations of the parameter values.

This new generation of our population is again evaluated and, thereby, closes the evolutionary circle. The overall process is iterated until we do not find any improvements for several generations in a row. At that point we stop the iteration and choose the fittest individuals as candidates for good parameter settings.

5 Experiments and Results

The prototype of our evolution framework was tested with some existing CE algorithms. In this section we describe the results we obtained in our optimisation experiments. As combinations of algorithms are not yet possible, we restrict ourselves to single algorithms. Therefore we will now take a very brief look at the employed algorithms and describe in particular how they are affected by changes in the parameters.

5.1 Link quota filter

Link Quota Filters (LQF) are an intuitive heuristic for identifying and removing link lists and navigational elements. Each block level element of the DOM tree¹ is analysed for the ratio of text used as link anchors (i.e. appearing in a element with a `href` attribute) to the overall text. Nested block level elements are not considered. If this ratio is above a certain threshold t the entire block is discarded as not being part of the main content. The value t is the only parameter for this CE algorithm². So far, LQF has been analysed for threshold ratios of 0.25, 0.5 and 0.75.

5.2 Content Code Blurring

The content code blurring (CCB) algorithm and its variations ACCB and TCCB (Gottron, 2008b) identify source code regions which correspond to an area of the document with a homogeneous layout. To detect such regions they calculate a weighted average ratio of code and content elements in local neighbourhoods. If the ratio is above a certain threshold, the neighbourhood is declared to contain a part of the main content. The size r of the neighbourhood and the threshold value t are the main parameters for this algorithm. In previous experiments, settings of $t = 0.75$ and $r = 40$ had been empirically found to perform well for ACCB, while TCCB had been used with values $t = 0.75$ and $r = 25$.

¹The DOM tree (Hors *et al.*, 2004) is a standardised representation of XML or HTML Documents. It is based on the inherent hierarchy of nested elements in such documents.

²There are variations of the algorithm (Mantratzis *et al.*, 2005), which introduce other parameters. However, also in these cases the link quota threshold affects the extraction performance most.

5.3 Stopword Filter

The observation that additional contents like navigation menus, link lists or commercials usually do not contain complete sentences but rather headwords, motivates stopword filters (SW). A stopword filter analyses the blocks of an HTML document for the amount of contained stop words, such as particles, conjunction and pronouns. In “regular” sentences they appear frequently to form syntactically correct sentences. In additional contents, instead, they appear far less. So, if the ratio of stop words is above a certain threshold t , SW considers the entire block to contain well formed language and accordingly to be part of the main content.

As the SW algorithm approaches the CE task from a computer linguistic point of view, it is not enough to provide a single threshold. Instead, the threshold depends heavily on the language of the content³. Our SW implementation is capable of recognising English, German and Italian documents and, accordingly, is equipped with three thresholds t_{EN} , t_{DE} and t_{IT} for the stop word ratio in each language.

In this case the parameter settings are entirely independent from each other. Hence, it is particularly well suited for our evolutionary approach, especially for the recombination of settings.

5.4 Experiments

As mentioned above we compiled a small set of test documents for the evolutionary framework. The documents and their gold standard for the main content were picked as a random, representative sample from the much larger evaluation sets used in earlier work. In this way we created a set of 120 documents, which remained the same throughout the entire optimisation process. A set of this size allows on one hand a quite quick⁴ evaluation of a parameter setting for an algorithm and on the other hand should provide quite reliable results to judge the overall fitness of an algorithm.

We started the optimisation process with an initial population composed of random parameter configurations and added an individual with the so far best performing parameter settings. So, for LQF we added an individual with a threshold value of 0.25, for ACCB and TCCB we set t to 0.75 and r to 40 and 25 respectively. For the SW filter we used settings of $t_{EN} = 0.3$, $t_{DE} = 0.25$ and $t_{IT} = 0.4$.

The evolutionary process spawned several configurations which yielded good results on the small test set. Table 2 lists the settings for the top 4 fittest individuals at the end of the process. Except for ACCB, we found new configurations which performed better – at least on the small test set.

Another interesting point are the parameter configurations which were tried out during the process. The plot in figure 4 shows very nicely how the search was focussed on an area, where finally also the best configuration was found. So, instead of exploring the full parameter space, our approach restricts exploration to promising parts of the configuration space.

To finally compare these configurations with previous settings for the parameters, we evaluated their performance on the full set of 9.601 evaluation documents

³For some languages, SW filters are actually impossible to use, as they hardly contain any

TABLE 2: Parameter values for the fittest individuals at the end of the evolutionary process.

LQF t	ACCB		TCCB		SW		
	r	t	r	t	t_{EN}	t_{DE}	t_{IT}
0.27	40	0.75	16	0.75	0.31	0.32	0.38
0.28	38	0.75	18	0.75	0.31	0.32	0.28
0.23	42	0.75	23	0.75	0.31	0.32	0.17
0.26	40	0.8	24	0.76	0.3	0.25	0.4

TABLE 3: Word Sequence F1

	bbc	chip	economist	espresso	golem	heise	manual	repubblica	slashdot	spiegel	telepolis	wiki	yahoo	zdf	Macro Avg.
ACCB ($r; t$)															
(40;.75)	0.924	0.703	0.890	0.875	0.959	0.916	0.419	0.968	0.177	0.861	0.908	0.682	0.847	0.929	0.790
(42;.75)	0.923	0.700	0.892	0.876	0.962	0.917	0.420	0.968	0.186	0.862	0.906	0.680	0.845	0.928	0.790
(38;.75)	0.924	0.705	0.888	0.875	0.955	0.916	0.418	0.967	0.176	0.860	0.908	0.686	0.850	0.931	0.790
(40;.80)	0.919	0.729	0.905	0.872	0.959	0.912	0.418	0.955	0.195	0.865	0.902	0.616	0.844	0.923	0.787
LQF (t)															
(.23)	0.834	0.502	0.739	0.667	0.943	0.792	0.387	0.826	0.136	0.789	0.906	0.671	0.801	0.579	0.684
(.25)	0.834	0.502	0.732	0.667	0.926	0.791	0.387	0.826	0.135	0.790	0.906	0.690	0.799	0.579	0.683
(.26)	0.834	0.502	0.732	0.667	0.903	0.790	0.387	0.826	0.133	0.790	0.907	0.701	0.795	0.579	0.682
(.27)	0.834	0.502	0.732	0.667	0.903	0.787	0.387	0.826	0.133	0.790	0.907	0.709	0.792	0.579	0.682
(.28)	0.834	0.502	0.731	0.667	0.902	0.787	0.387	0.826	0.133	0.790	0.906	0.717	0.791	0.579	0.682
SW (t_{EN}, t_{DE}, t_{IT})															
(.31;.32;.38)	0.771	0.546	0.867	0.872	0.770	0.849	0.412	0.909	0.106	0.749	0.926	0.616	0.637	0.846	0.705
(.31;.32;.28)	0.771	0.546	0.867	0.808	0.770	0.849	0.412	0.869	0.106	0.749	0.926	0.616	0.637	0.846	0.698
(.30;.25;.40)	0.741	0.544	0.776	0.875	0.770	0.832	0.412	0.837	0.106	0.739	0.926	0.629	0.637	0.848	0.691
(.31;.32;.17)	0.771	0.546	0.867	0.670	0.770	0.849	0.412	0.817	0.106	0.749	0.926	0.616	0.637	0.846	0.684
TCCB ($r; t$)															
(18;.75)	0.919	0.886	0.871	0.873	0.876	0.885	0.416	0.924	0.208	0.872	0.904	0.692	0.873	0.841	0.789
(23;.75)	0.916	0.866	0.902	0.870	0.945	0.845	0.404	0.921	0.259	0.901	0.903	0.670	0.876	0.757	0.788
(24;.76)	0.911	0.837	0.903	0.869	0.946	0.887	0.405	0.918	0.268	0.909	0.897	0.652	0.871	0.718	0.785
(25;.75)	0.914	0.842	0.903	0.871	0.947	0.821	0.404	0.918	0.269	0.910	0.902	0.660	0.873	0.745	0.784
(16;.75)	0.920	0.755	0.870	0.874	0.878	0.886	0.417	0.922	0.183	0.863	0.905	0.701	0.874	0.871	0.780

mentioned earlier (see table 1). The results of this evaluation are shown in table 3, where the better performing versions are listed first. For comparison we also included the configurations previously known to perform best, if they were not a result of the evolutionary process anyway.

For LQF, SW and TCCB it can be seen, that the performance has improved, for ACCB equivalent configurations have been found. The improvements for LQF and TCCB are not very big and can rather be considered as finetuning. This is mainly due to the fact, that the algorithms had already been analysed for good

stop words.

⁴Depending on the complexity of the CE algorithm, evaluation of a set of parameters took usually between 1 and 2 minutes.

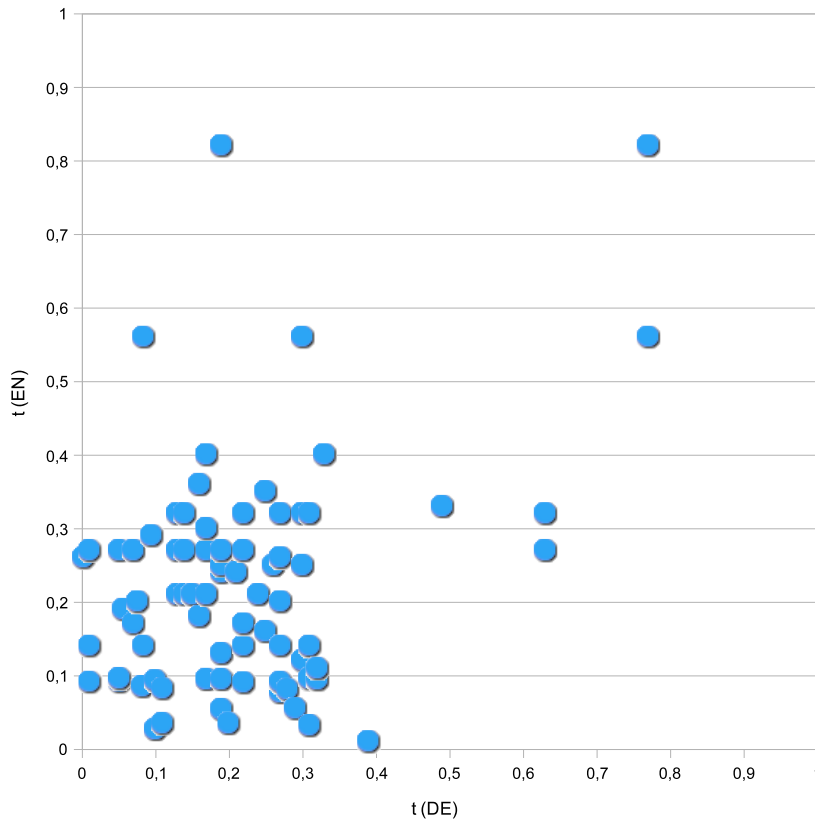


FIGURE 4: Distribution of parameters t_{DE} and t_{EN} while searching for optimal configuration of the SW algorithm.

parameter settings in earlier work – even if not as methodical as here. However, the small increases in average F1 show, that there was still some potential for improvements. In this way, the performance of TCCB nearly caught up with ACCB.

In the case of the SW algorithm the improvements are significantly larger. This might be because the algorithm had not undergone a thorough analysis in the configuration space, but also because the mutually independent parameters are quite suitable for recombination of good results.

In general we can say, that the evolutionary approach to parameter optimisation works fine and helps to improve performance of CE algorithms. The improvements in the small test set were reflected very well on the much larger collection of documents we used for the final evaluation.

The option to exchange the test set documents in order to optimise the parameters for different document styles, e.g. with particularly long, short or fragmented main contents, is also interesting. The potential of such an approach can be seen in

the different performances of the settings of ACCB on the single packages. While the average F1 does not change, F1 in the single packages shows changes. For some packages – i.e. documents scenarios – the values increased, for others they decreased. If the user knows what kind of documents he has to expect in his CE application, we can choose suitable test documents accordingly.

6 Conclusions and Future Work

In this paper we demonstrated how evolutionary computing can be used effectively to improve parameter settings of content extraction algorithms. We incorporated the feedback of an established evaluation system as fitness function and used populations of different parameter settings. Experiments with an implemented prototype proved the approach to be suitable for optimising parameter settings. The freedom and independence of a particular trainingset allows to optimise CE algorithms also for very specific and individual scenarios.

In the context of the mentioned CombinE system (Gottron, 2008a), one issue is to extend the optimisation beyond parameter settings of a single algorithm to the possibilities of how to combine different CE filters. As a core feature of CombinE is to create complex pipelines of different CE filter modules, optimisation has to be extended to the order and the structure in which the filters are applied. Also the recombination process of the fittest individuals might be modelled more detailed. Semantically independent parameters, e.g. the threshold values for different languages in SW, are suitable for random recombination of parameter values from the individuals. Correlated parameters, e.g. threshold and neighbourhood range in ACCB, instead, might benefit if the values are combined in a more elaborated way, say, by calculating the mean value. In general, our prototype could be extended to control the parameters of the evolutionary process itself, e.g. change how many individuals to use for the parent generation or adapt the mutation rate depending on the current developments in the individuals' fitness. Another interesting question is how to model the fitness function in order to optimise recall or precision using restrains on the respectively other performance value.

References

- Sandip DEBNATH, Prasenjit MITRA, and C. Lee GILES (2005), Identifying Content Blocks from Web Documents, in *Foundations of Intelligent Systems*, Lecture Notes in Computer Science, pp. 285–293.
- Aidan FINN, Nicholas KUSHMERICK, and Barry SMYTH (2001), Fact or Fiction: Content Classification for Digital Libraries, in *DELOS Workshop: Personalisation and Recommender Systems in Digital Libraries*.
- Thomas GOTTRON (2007), Evaluating Content Extraction on HTML Documents, in *ITA '07: Proceedings of the 2nd International Conference on Internet Technologies and Applications*, pp. 123–132, ISBN 978-0-946881-54-3.
- Thomas GOTTRON (2008a), Combining content extraction heuristics: the CombinE system, in *iWAS '08: Proceedings of the 10th International Conference on Information Integration and Web-based Applications & Services*, pp. 591–595, ACM, New York, NY, USA, ISBN 978-1-60558-349-5, doi:<http://doi.acm.org/10.1145/1497308.1497418>.

- Thomas GOTTRON (2008b), Content Code Blurring: A New Approach to Content Extraction, in *DEXA '08: 19th International Workshop on Database and Expert Systems Applications*, pp. 29 – 33, IEEE Computer Society.
- Suhit GUPTA, Gail KAISER, David NEISTADT, and Peter GRIMM (2003), DOM-based Content Extraction of HTML documents, in *WWW '03: Proceedings of the 12th International Conference on World Wide Web*, pp. 207–214, ACM Press, New York, NY, USA, ISBN 1-58113-680-3, doi:<http://doi.acm.org/10.1145/775152.775182>.
- D. S. HIRSCHBERG (1975), A linear space algorithm for computing maximal common subsequences, *Commun. ACM*, 18(6):341–343, ISSN 0001-0782, doi: <http://doi.acm.org/10.1145/360825.360861>.
- Arnaud Le HORS, Philippe Le HÉGARET, Lauren WOOD, Gavin NICOL, Jonathan ROBIE, Mike CHAMPION, and Steve BYRNE (2004), Document Object Model (DOM) Level 3 Core Specification, W3C Recommendation, URL <http://www.w3.org/TR/2004/REC-DOM-Level-3-Core-20040407/>.
- Constantine MANTRATZIS, Mehmet ORGUN, and Steve CASSIDY (2005), Separating XHTML content from navigation clutter using DOM-structure block analysis, in *HYPERTEXT '05: Proceedings of the sixteenth ACM conference on Hypertext and hypermedia*, pp. 145–147, ACM Press, New York, NY, USA, ISBN 1-59593-168-6, doi: <http://doi.acm.org/10.1145/1083356.1083384>.
- Zbigniew MICHALEWICZ (1996), *Genetic Algorithms + Data Structures = Evolution Programs*, Springer, 2nd edition.
- David PINTO, Michael BRANSTEIN, Ryan COLEMAN, W. Bruce CROFT, Matthew KING, Wei LI, and Xing WEI (2002), 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*, pp. 46–55, ACM Press, New York, NY, USA, ISBN 1-58113-513-0, doi:<http://doi.acm.org/10.1145/544220.544228>.
- A. F. R. RAHMAN, H. ALAM, and R. HARTONO (2001), Content extraction from HTML documents, in *WDA 2001: Proceedings of the First International Workshop on Web Document Analysis*, pp. 7–10.
- Tim WENINGER and William H. HSU (2008), Text Extraction from the Web via Text-Tag-Ratio, in *TIR '08: Proceedings of the 5th International Workshop on Text Information Retrieval*, pp. 23 – 28, IEEE Computer Society.