EVALUATING CONTENT EXTRACTION ON HTML DOCUMENTS

Thomas Gottron

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

ABSTRACT

A variety of applications uses methods to determine and extract the main textual contents of an HTML document. The performance of the methods employed in this task is rarely evaluated. This paper fills this gap by introducing a platform independent and extensible framework for measuring, evaluating and comparing the performance of methods for Content Extraction. We further give an overview over extraction algorithms found in domain specific applications and present an adaptation of a related algorithm to perform Content Extraction. We compare the algorithms using the developed framework and show that our adapted algorithm performs best on most HTML documents.

KEYWORDS

Content Extraction, Evaluation Measures, Web Mining

1. INTRODUCTION

Content Extraction (CE) is the process of determining the parts of an HTML document which contain the main textual content of this document. A human user nearly naturally performs some kind of Content Extraction when reading a web page by ignoring the parts with additional contents, such as navigation, functional and design elements or commercial banners − at least as long as they are not of interest. Gibson, Punera and Tomkins [1] estimated those additional and usually uninteresting contents to make up around 40 to 50% of most web pages on the Internet.

Though it is a relatively intuitive task for a human user, it turns out to be difficult to determine the main content of a document in an automatic way. Several approaches deal with the problem under very different circumstances. For example, CE is used extensively in applications rewriting web pages for presentation on small screen devices or access via screen readers for visually impaired users. Some applications in the fields of Information Retrieval (IR), Web Mining and Text Summarisation use CE to pre-process the raw data in order to improve accuracy. It becomes obvious that under the mentioned circumstances the extraction has to be performed by a general approach rather than a tailored solution for one particular set of HTML documents with a well known structure.

As CE often represents only a part of those applications, it is rarely evaluated on its own. Moreover, to our knowledge the methods used to extract the main content are hardly ever compared to each other directly. We fill this gap by introducing a framework for measuring, evaluating and comparing the performance of CE methods based on objective criteria. The framework is designed to be platform independent and extensible for both, the evaluation data and the measures to be employed. It has been successfully used to compare CE algorithms we found in domain specific works to demonstrate its functionality. The results of this evaluation and an overview of extraction methods is part of this paper.

We proceed as follows. In section 2 we describe the concept of Content Extraction. We develop several measures for evaluating the performance of a CE method in section 3 and in 4 we outline the architecture of the evaluation framework. In 5 we give a short overview over algorithms for CE and compare their performance. We finally describe related works in Section 6 and conclude the paper with some remarks on the framework and a discussion of future extensions.

2. Content Extraction in HTML Documents

As explained above we define Content Extraction as the process of determining the main content in an HTML document, and conversely identifying the parts which contain additional contents, such as navigation elements or commercial banners. So, Content Extraction can after all be seen as a function from an HTML document to a set of pairwise indices determining at which position in the string representation of the document the parts of the main content start and end. We formally define CE as a function f_e with:

$$f_e : D \alpha \ \{(s_i, e_i) | 0 \leq s_i \leq e_i \leq |D|, i \in I\}$$

where D is a document represented by $|D|$ characters, s_i and e_i are the start and end position of the first and respectively last character of a part of the main content, with i out of an finite indexing set I allowing to enumerate those parts of a document.

Accordingly an algorithm for CE will have to determine the index-tuples and use them to outline the main content. An outline in the form of the indices itself will rarely occur, considering the environment of applications we have mentioned. Instead solutions will usually use the indices to remove additional contents from the HTML document, wrap them in comments or replace them with whitespace. Adding additional markup or extracting and transforming the main content into plain text are other possible solutions.

Particularly this last approach suggests an interesting interpretation of CE as a task of IR within a single document. After all, CE tries to find the relevant parts of the document for the query "What is the main content?". We keep this interpretation in mind while developing the evaluation measures.

3. Measures for Evaluating Content Extraction

To be able to measure the performance of a CE algorithm on a particular document, we first of all need an independent expert to define what exactly is the main content of this document and thereby to create a gold standard (even though this question of what is or is not part of the main content may be difficult to answer and is prone to subjectivity, we rely on our expert to have at least a good guess on what parts of the document make up the main content). Given such a standard, we can compare it with the output of the CE algorithm − we will refer to this output as the extracted content to keep it distinguishable from the main content provided by the expert.

A primary objective in an evaluation process must be to measure how similar the extracted content is to the expert's notion of the main content. Clearly large accordance should be rewarded by an according measure, while both possible deviations − extracting too much or extracting too little − should be penalised.

Interpreting the extraction procedure again as an IR task allows us to use the concepts of precision and recall for this purpose. The precision P is the ratio of extracted relevant items to all extracted items, the recall R instead is the ratio of extracted relevant items to all relevant items. Based on those two concepts, the F1-measure (defined as $(2 \cdot P \cdot R)/(P+R)$) provides a suitable similarity measure to compare the extracted items with the relevant items.

To adapt precision and recall to fit our intended document level evaluation we have to define what are the retrievable items in an extraction process. We will use different concepts for those items resulting in slightly different measures providing different levels of granularity:

1. Characters: Each character is a fragment of the document and as such can be part of the relevant main content or an irrelevant additional content. This approach might be useful if not only letters or digits are examined, but if as well punctuation marks, other particular symbols and word length are supposed to influence the measures result.

2. Words as a sequence: The text content is tokenised into words. Based on the word tokens as items we can again decide, whether or not it was a word from a relevant part.

3. Words as a bag of words: A bag of words is a vector of words, counting how many times they do appear. This document representation is of interest as it is used for several text classification methods and implies that the order of words does not matter.

4. Words as a set: Opposite to the bag of words approach it is no longer of interested how often a word appears. We simply check for the presence or absence of a word in main and extracted content.

Beside a high score regarding the F1 measures, a method should be as well stable in the sense, that it at least performs similar on similar documents. For this we take a look at the estimated standard deviation of F1 over a sample of documents. A low standard deviation points to a steady and stable performance, while a high value indicates fluctuations in the methods performance.

As a last measure we consider the time needed to process a document. If the extraction process is made on the fly (e.g. by passing the web pages through a proxy server) the user may not be willing to wait too long for the results. As the document length will certainly matter we brake down the processing time to seconds per kB. Even though the methods may not operate on a byte level, it is the most neutral dimension to normalise the processing time for documents with different length. However, the assumption does not seem too far fetched, that the number of HTML elements, tags, words or whatever other structures are used as basis for the extraction process increase more or less linear with the byte length of the document.

4. Framework Design

The evaluation framework comprises three parts. First we define a simple data format to store the test data, including documents, meta data and the main content as determined by an expert. Second a platform independent architecture allowing evaluation of virtually all HTML document based CE methods. And finally an output format for the results of the evaluation, which can easily be used by other applications for further processing.

4.1. Test Packages

A set of test data must comprise several information. The most obvious of which is the raw HTML document the extractor has to operate on and the definition of the main content provided by an expert. For the last purpose we chose to use plain text as it improves readability compared to the indexing outline introduced in the formal definition. To allow storing some additional knowledge about the web page we permit to attach meta information. So far we store the URL of the original article, the language and two flags whether the main content is fragmented (i.e. is interrupted by additional contents) and whether the HTML document contains multiple articles.

We organise the test data in test packages, to allow compiling test series for different scenarios. A test package simply consists of a description file and the files containing the HTML documents, their main content as text and their meta information. To link a document with its additional files, they share the same name using solely different extensions. This simple file based format allows easy recompilation and adaptation of test packages, if necessary even manually.

4.2. Architecture

Given the fact, that we are working with web documents, the approach to use web techniques for realizing the platform independence is nearly a self-evident solution. Some of the existing applications for CE already come along as a proxy server, others can easily be wrapped into such a server. This concept separates the framework entirely from the CE implementations, guaranteeing a maximum freedom on both sides.

To run a test on a document, we start a minimal web server thread, which serves exactly the one document to undergo testing, before it shuts down itself. In a next step the framework issues the proxy capable of Content Extraction to retrieve (and implicitly process) the document from the previously launched server. During this process the time between the completion of serving the document from the web server till the completion of receiving it from the proxy is measured to estimate the time needed for the extraction.

As the proposed evaluation measures are based on unstructured text, we can − once the document has been returned by the proxy − strip off all tags, comments, style and script elements. After resolving HTML entities into Unicode characters and reducing superfluous whitespace (As cumulative whitespace is ignored in the presentation of HTML documents it is usually used to layout the source code. As such it does not affect the content and accordingly we ignore it.) we have a string representation of the extracted content, equivalent to the plain text format of the experts main content. For the word based measures we further split both the extracted and the main content into word tokens.

To calculate precision and recall it is essential to find the intersection of the extracted and the main content for defining the extracted relevant elements. We determine this intersection for the methods based on characters and the word sequence by calculating the longest common subsequence using the algorithm of Hirschberg [2]. The intersection for the measures based on a set of words is the natural intersection of sets; for the measure using bag of words we take the minimum number of occurrences of a word in the extract and the main content.

4.3. Output Format

Running an evaluation on a complete test package produces the above mentioned values for processing time, precision, recall and F1-measure for each document in the package plus the standard deviation of F1 over all documents. For storage we decided again for a simple and platform independent format to permit further processing of the results. We save the results of each run in a CSV file using tabulators to separate the values.

5. Performance of Extraction Algorithms

Few algorithms have been developed to an extent, that they are ready to be used. Some of the papers we found dealing with CE remain vague about implementing the proposed concept, even less provide an implementation. Several approaches (e.g. in [3-7]) present general algorithms to build specialised extractors for a set of web documents. They can not be used out of the box on an arbitrary HTML document and as such lack the generality we demanded.

We have collected some extraction algorithms from various fields of application which are well described and are designed to be generally applicable to all kinds of documents without prior adjustments. We used our framework to evaluate and compare their extraction performance.

As a baseline in the evaluation process we used a proxy which serves the documents unchanged. This “plain”' method for CE will always score a perfect recall, but will perform poor considering the precision.

5.1. Crunch

One of the well developed approaches is Crunch introduced by Gupta et al. in [7] and refined in [8-10]. The main objective of Crunch is to optimise an HTML document for displaying it on small screen devices or to improve accessibility for screen readers. It therefore uses a collection of heuristics to filter the content, e.g. by discovery of link lists, text lists, blocks with a too high link ratio, commercial banners, etc. We used the latest version 2.0 with standard settings.

5.2 Body Text Extraction

Another solution was introduced by Finn et al. in [11]. As a pre-processor for their application they developed the Body Text Extraction (BTE) algorithm. The algorithm is based on the assumption, that the main content is a single continuous block of text containing few or no HTML tags. To find this text block BTE represents a document as a sequence of *N* tokens B_n. Each HTML tag is a token with a value of 1 and each word in the text is a token with a value of 0. The extraction is formulated as an optimisation problem: to find two tokens *i* and *j* in the token sequence of a document, for which the number of text tokens between *i* and *j* and the number of tag tokens outside *i* and *j* is maximal. This corresponds to maximising $T_{i,j}$ where:

$$T_{i,j} = \sum_{n=0}^{i-1} B_n + \sum_{n=i}^{j} (1 - B_n) + \sum_{n=j+1}^{N-1} B_n$$

The BTE algorithm itself has not been tested, but it seemed to perform well in the application it was designed for. Although there exists a reference implementation in Python, we re-implemented the algorithm in Java for easier incorporation into an existing proxy server.

5.3. Document Slope Curve

The data structure underlying BTE of text and tag tokens is used as well in [12] to construct a Document Slope Curve (DSC) function *d* counting the number of tag tokens appearing up to a certain point (in terms of tokens) in the document:

$$d(i) = \sum_{n=0}^{i} B_n \quad for \quad 0 \le i \le N-1$$

To overcome the restriction of BTE to discover a single continuous block of text only, the idea is to look for plateaus in this curve with a low slope. Each found plateau represents a larger text block with few tags in the document and as such is assumed to be a part of the main content.

The first step to discover those plateaus is to pass a window with 50% overlap and a fixed token length *l* over the document and to look for low slope windows. A window starting at token *i* is declared to have a low slope, if it has an average slope which is less than half the average slope of the entire document. Three consecutive low slope windows indicate the beginning of a low slope area, until three consecutive windows with a slope higher than half the average document slope are found.

The original application was not intended to perform Content Extraction, but used the overall rate of low slope areas for distinguishing between documents which do or do not contain articles. Therefore we implemented a new variation which extracted the low slope areas.

5.4. Link Quota Filter

Additionally we implemented a Link Quota Filter (LQF), an intuitive heuristic for identifying and removing link lists and navigational elements. Each block level element of the DOM tree is analysed for the ratio of text used as link anchors (i.e. in a elements) to overall text. Nested block level elements are not considered. If this ratio is above a certain threshold the entire block is discarded as not being part of the main content. As threshold we used ratios of 0.25, 0.5 and 0.75. LQF is used with slight variations and adaptations by Mantratzis et al. [13] and as part of Guptas Crunch [8].

5.5. Test Packages

To generate test packages, we had to collect HTML documents and have an expert determine their main content. Employing a human user allows the creation of very diversified test packages, as a human can outline quite easily the main content of a document (within certain boundaries due to subjectivity). To support the user in his task, we implemented an extension to Mozillas Firefox browser, allowing to save an HTML document together with its meta information and the selected text of the main content.

With this human expert approach we created a collection of 65 documents from a variety of websites with different backgrounds. Including e-commerce, online newspapers with short news flashes and full articles, manuals and references, the collection covers a wide range of different web pages containing textual main contents. To reduce outliers caused by extreme forms of single articles, we took five pages from each website, if possible from different topics.

We additionally implemented specialised extractors for particular web sites. From those we downloaded several documents and extracted the main content to compile test packages for each site. These automated packages allow massive testing on documents with different contents but similar structure, thus providing a good base for evaluating the stability of CE methods.

With this approach we generated test packages with articles from the online news sites of ZDF heute (www.heute.de, 263 documents), The Economist (www.economist.com, 229 documents), Spiegel (www.spiegel.de, 680 documents), heise online (www.heise.de, 487 documents) and Slashdot (www.slashdot.org, 398 documents), the online articles of the Telepolis magazine (www.telepolis.de, 176 documents) and a random selection of Wikipedia articles (de.wikipedia.org, 497 documents). Except for the economist and the slashdot package all the pages in the automatically generated packages were in German language.

5.6. Results

We passed all the packages through all the CE algorithms, comparing their performance within each package. The results for the F1-measures are shown in the charts in figure 1. The character based measure is labelled “Str f1”', the word sequence “Seq f1”', the bag of words measure with ”Vec f1”' and the set as “Set f1'”.

With exception of the wiki and telepolis package, the results are similar considering the relative performance. DSC scores highest in all the F1-measures, followed by Crunch. The results of the LQF algorithm vary surprisingly little for the different threshold settings. LQF and BTE are in general comparable, slight differences depend on individual advantages on particular document structures.

The wiki package turns out to be very tricky for all the algorithms and concerning the F1-measures it seems best not to use any CE method at all, as the values for plain extraction are the highest. Taking into account, that basically all methods try to identify parts in a document with little links or long texts with no tags, it becomes obvious, that a wiki article with its usually high rate of in-text-links is the worst scenario possible. The telepolis package instead provides the opposite case. Long text, few or no hyperlinks, little markup in the text. Accordingly all methods perform very well.

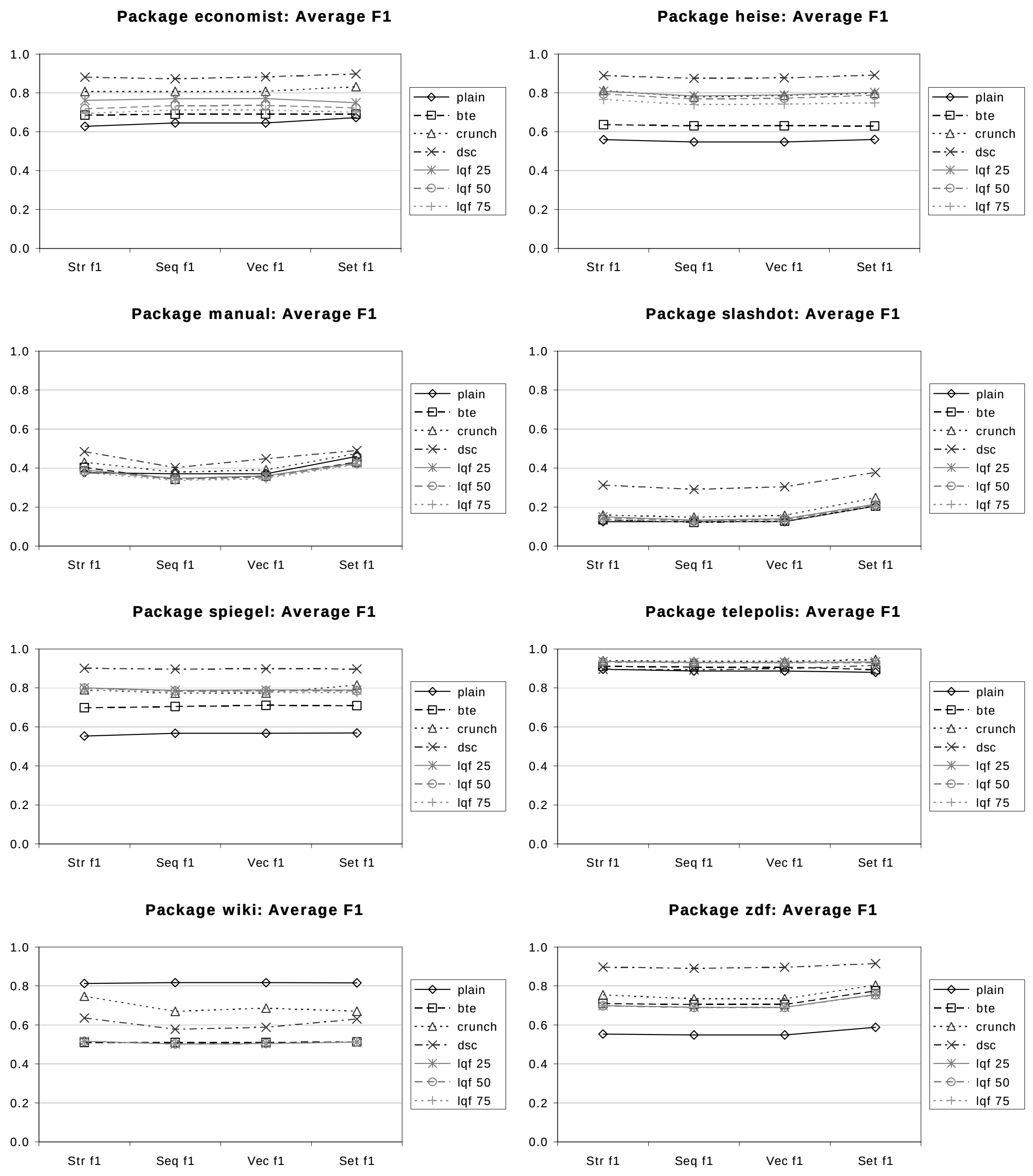

Figure 1. Average F1 performance per package

When choosing a CE algorithm for a particular application for which the impact of missing or superfluous data is known, it is as well interesting to take a closer look at precision and recall separate. We show the results in figure 2 for the heise and the wiki package, as heise is quite representative for most other packages and wiki a special case.

The charts clearly show the tradeoff between precision and recall. A method scoring high in recall usually does poor for precision, which is nothing new in the field of IR. However, it seems that with little loss in recall the methods usually achieve a quite good improvement in the precision rating. The problem of the methods with the wiki package becomes obvious at this point. While precision improves only little or even decreases for some algorithms, the recall rate drops drastically.

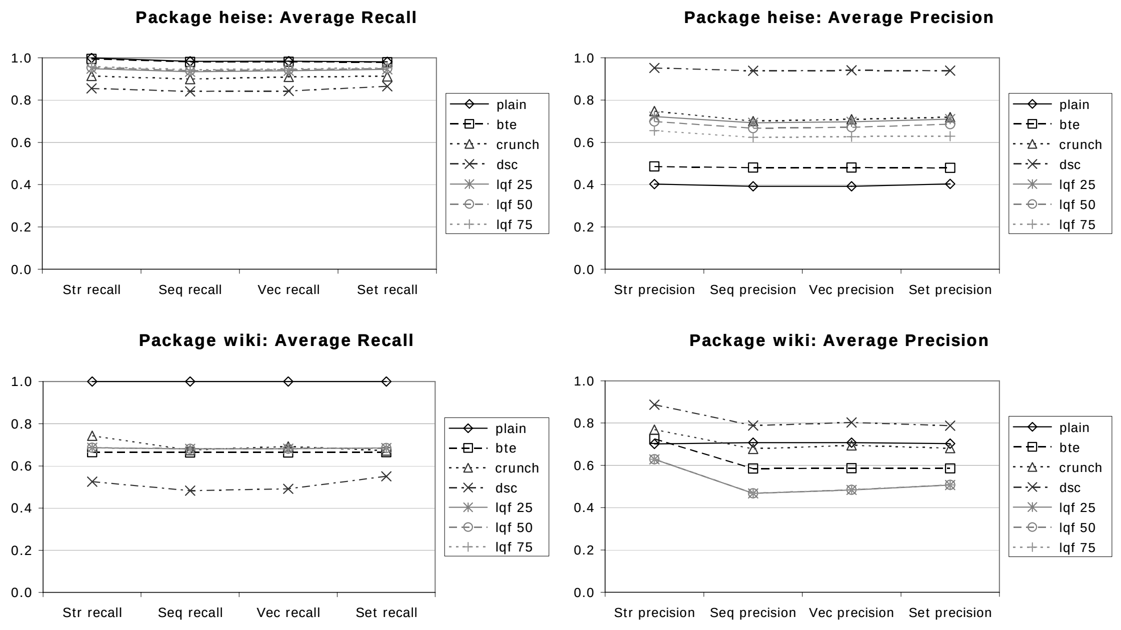

Figure 2. Average precision and recall for the heise and wiki packages

Figure 3 shows the processing time and estimated F1 standard deviation for spiegel as a representative and as well the largest package. As in all automatically extracted packages the standard deviation roughly ranges from 0.1 to 0.15 for all methods. Looking at the time performance of the algorithms, DSC is the fastest, though the differences with other methods are negligible. Only BTE peeks out, by taking much longer on average to process a document. Finding the solution to the optimisation problem turns out to be expensive, rendering BTE unsuitable for use in an on-the-fly system.

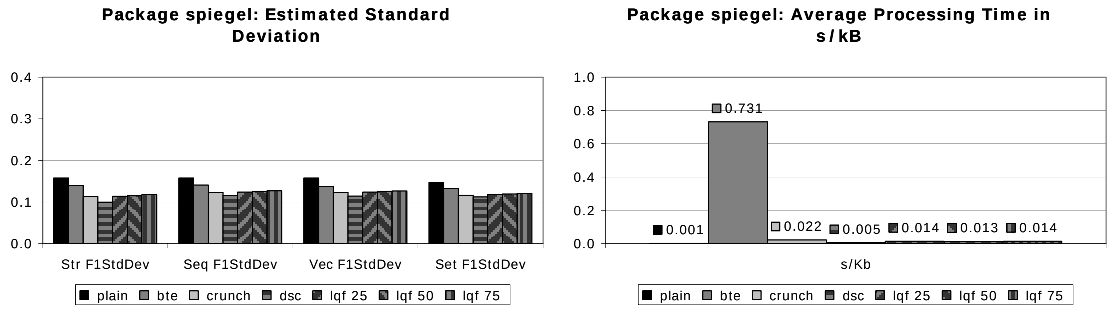

Figure 3. Estimated F1-measure standard deviation and processing time for the spiegel package

6. Related Work

To our knowledge few works cover methods for extracting unstructured content such as news articles from HTML documents. Accordingly even less work is done about measuring the performance of these methods.

Rahman et al. list in [14] requirements a CE system for HTML documents should comply with. Being generic enough to work with any website, a fast extraction algorithm and a non intrusive design are the aspects they consider most important for the extraction part of such a system.

Most similar to our approach for evaluation are the works of Debnath et al. [15,5] and of Lin and Ho [2]. They use precision and recall based on finding blocks as items for retrieval, which suits their methods of extracting block structures. However, not all main contents are perfectly alignable with those block structures and the measure does not seem to take into account the "amount" of content present in a block. Under this measure a block with no content is considered as important as a block containing long texts.

Yi et al. [4] introduce Site Style Trees to find and remove additional contents. They evaluate the extraction indirectly by looking at improvements in classification and clustering of documents.

Gupta [16] has developed four measures for evaluating the performance of the Crunch framework, two of which reflect the intended application to improve accessibility of web pages to screen readers and small screen devices. The Screen Reader Testing compares the seconds needed to read the output of the Content Extraction, the Constraint Device Testing counts the words displayed on small screen devices. More general, the Performance and Scalability Testing evaluates the systems time performance under heavy load. The last measure is the Newsblaster Testing and evaluates the impact of Crunch on the news tracking and summarisation system Newsblaster. An improvement is achieved in the sense, that the amount of web pages not suitable for the summarisation process is reduced drastically thanks to the prior Content Extraction.

Another common approach is to employ human users in testing extraction systems. This may be done by supervising the users completing certain tasks (like finding a particular information) faster or with less user interactions as in [17]. Alternatively the users can be asked directly to evaluate the performance of an extraction system as in [18]. However, human evaluation has several flaws: It is expensive, time consuming, prone to subjectivity and cannot be automatised.

Precision, recall and F1-measure are standard measuring techniques in the field of IR. Introductory literature like [19] may give a good overview to this topic. In combination with gold standards and on a document level this measures are used as an extrinsic evaluation measure for text summarisation, e.g. in [20].

7. Conclusions and Future Work

We presented a framework for measuring and comparing Content Extraction methods in an objective way. The framework has been successfully used to evaluate various extraction algorithms, allowing for the first time a straight comparison of those algorithms. As a result of this comparison we can declare DSC with our adaptations to be the best performing method, followed by Crunch. However a general recommendation for a method can not be given as the intended frame application may impose particular requirements concerning precision or recall.

In the future we plan to extend the framework in several directions. Beside extending and widening the test packages with more and different HTML documents, we intend to introduce additional measures. Given the application of CE as a preparatory step for document classification, we want to measure the impact of an extraction method on classifiers. For this purpose it might be as well interesting to see the development of precision, recall and F1-measure if NLP techniques like stop word reduction and stemming are used on the word based structures of sequence, bag of words and set. Finally we keep on searching for other extraction algorithms to evaluate and compare their results with the ones presented in this paper.

References

[1] 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*. 2005, pp. 830–839.

[2] D. S. Hirschberg, "A linear space algorithm for computing maximal common subsequences," *Commun. ACM*, vol. 18, no. 6, pp. 341–343, 1975.

[3] 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*. 2002, pp. 588–593.

[4] 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*. 2003, pp. 296–305.

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

[6] H.-Y. Kao, J.-M. Ho, and M.-S. Chen, “Wisdom: Web intrapage informative structure mining based on document object model,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 5, pp. 614–627, 2005.

[7] D. C. Reis, P. B. Golgher, A. S. 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*. 2004, pp. 502–511.

[8] 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*. 2003, pp. 207–214.

[9] S. Gupta, G. E. Kaiser, P. Grimm, M. F. Chiang, and J. Starren, “Automating content extraction of html documents,” *World Wide Web*, vol. 8, no. 2, pp. 179–224, 2005.

[10] 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*. New York, 2006, pp. 875–876.

[11] 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.

[12] 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*. 2002, pp. 46–55.

[13] 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*. 2005, pp. 145–147.

[14] A. F. R. Rahman, H. Alam, and R. Hartono, “Content extraction from html documents,” in *WDA2001: Proceedings of the First International Workshop on Web Document Analysis*, 2001, pp. 7–10.

[15] 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*. 2005, pp. 1722–1726.

[16] S. Gupta, “Context-based content extraction of html documents,” Ph.D. dissertation, Columbia University, 2006.

[17] O. Buyukkokten, H. Garcia-Molina, and A. Paepcke, “Seeing the whole in parts: text summarization for web browsing on handheld devices,” in *WWW ’01: Proceedings of the 10th international conference on World Wide Web*. 2001, pp. 652–662.

[18] Y. Chen, W.-Y. Ma, and H.-J. Zhang, “Detecting web page structure for adaptive viewing on small form factor devices,” in *WWW ’03: Proceedings of the 12th international conference on World Wide Web*. 2003, pp. 225–233.

[19] I. H. Witten, A. Moffat, and T. C. Bell, *Managing Gigabytes : Compressing and indexing documents and images*, 1st ed. New York: van Nostrand, 1994.

[20] D. Marcu, “The automatic construction of large-scale corpora for summarization research,” in *SIGIR ’99: Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*. 1999, pp. 137–144.