More Effective Boilerplate Removal—the GoldMiner Algorithm

István Endrédy, Attila Novák

***Abstract*—The ever-increasing web is an important source for building large-scale corpora. However, dynamically generated web pages often contain much irrelevant and duplicated text, which impairs the quality of the corpus. To ensure the high quality of web-based corpora, a good boilerplate removal algorithm is needed to extract only the relevant content from web pages. In this article, we present an automatic text extraction procedure, GoldMiner, which by enhancing a previously published boilerplate removal algorithm, minimizes the occurrence of irrelevant duplicated content in corpora, and keeps the text more coherent than previous tools. The algorithm exploits similarities in the HTML structure of pages coming from the same domain. A new evaluation document set (CleanPortalEval) is also presented, which can demonstrate the power of boilerplate removal algorithms for web portal pages.**

***Index Terms*—Corpus building, boilerplate removal, the web as corpus.**

I. The Task

WHEN constructing corpora from web content, the extraction of relevant text from dynamically generated HTML pages is not a trivial task due to the great amount of irrelevant repeated text that needs to be identified and removed so that it does not compromise the quality of the corpus. This task, called boilerplate removal in the literature, consists of categorizing HTML content as valuable vs. irrelevant, filtering out menus, headers and footers, advertisements, and structure repeated on many pages.

In this paper, we present a boilerplate removal algorithm that removes irrelevant content from crawled content more effectively than previous tools. The structure of our paper is as follows. First, we present some tools that we used as baselines when evaluating the performance of our system. The algorithm implemented in one of these tools, jusText, is also used as part of our enhanced boilerplate removal algorithm. This is followed by the presentation of the enhanced system, called GoldMiner, and the evaluation of the results.

II. Existing Tools

In this section, some relevant boilerplate removal algorithms are reviewed, which are freely accessible and thus could be used as evaluation baselines. They contain good ideas, and the path of these good ideas are outlined in the following overview: methods often built on the result of the previous ones. We reimplemented some of these algorithms in C++, so that they can be evaluated in a fast and comfortable way.

Manuscript received on July 31, 2013; accepted for publication on September 30, 2013.

The authors are with the MTA-PPKE Language Technology Research Group and Pázmány Péter Catholic University, Faculty of Information Technology and Bionics, 50/a Práter street, 1083 Budapest, Hungary (e-mail: {endredy.istvan.gergely, novak.attila}@itk.ppke.hu).

A. The Body Text Extraction (BTE) Algorithm

The basic insights underlying the BTE algorithm [1] are the following:

1) the relevant part of the HTML content is usually a contiguous stretch,
2) the density of HTML tags is lower in it than in boilerplate content.

Based on these two assumptions, the algorithm performs a search for the longest stretch of text in which the number of intervening tags is minimal. The idea is simple, but the result is often wrong with the algorithm failing to extract the most relevant part of the content in situations where, contrary to the tag density assumption, it contains a segment with a higher tag-to-text ratio. This occurs, for example, if tables are included or advertisements interrupt the article. In this case, a significant part of the valuable content (or the whole) may be lost or replaced by entirely irrelevant content.

B. The Boilerpipe Algorithm

A merit of the boilerpipe [2] algorithm is that its authors demonstrated experimentally that boilerplate content can be identified effectively by using a good combination of simple text properties. They used an annotated training corpus of 500 documents (mainly Google news) to find the most effective feature combination. They tried to extract articles with the help of shallow text features, using 8-10 different feature combinations, and then they evaluated their results. In their experiments, a combination of word and link density features gave the best results (its F-measure was: 92%). Furthermore, the method is very fast and it needs no preprocessing. Both the training set and the tool can be downloaded.

C. The jusText Algorithm

The jusText algorithm [3] splits HTML content into paragraphs at block-level tags that are generally used to partition HTML content into logical units, such as `<p>`, `<td>`, `<h1>` etc. Using various features of these blocks of text such as the number of links (an idea from boilerpipe [2]), words and stopwords, the algorithm performs a rule-based classification of the blocks using various thresholds and a language-dependent list of function words tagging each unit 'good', 'almost good', 'bad', or 'too short'. The latter tag applies to units too short to categorize reliably. After initial classification, 'almost good' and 'too short' units surrounded by 'good' ones are reclassified as 'good'. The text to be extracted consists of all units classified as 'good' in the final classification. The algorithm performs quite well even for extreme pages.

However, inspection of the corpus generated by using the jusText algorithm to filter crawled news portals revealed that many expressions that obviously come from a single article and should not occur more than once, like *The feeding-bottle is a potential source of hazard*, were still extremely strongly over-represented. Examples in Table I are from a corpus crawled from Hungarian news portals applying jusText as a boilerplate removal tool.

TABLE I
EXAMPLES OF PHRASES OVERREPRESENTED DUE TO INADEQUATE BOILERPLATE REMOVAL

Phrase	Occurr.
Utasi Árpi-szerű mesemondó. 'Utasi Árpi-like storyteller.'	10,587
A cumisüveg potenciális veszélyforrás. 'The feeding-bottle is a potential source of hazard.'	1,578
Obama amerikai elnök, 'U.S. President Obama,'	292
etióp atléta: cseh jobbhátvéd 'Ethiopian athlete: Czech right-back'	39,328
Barack Obama amerikai elnök 'U.S. President Barack Obama'	2,372
George Bush amerikai elnök 'U.S. President George Bush'	1,626

We found that the problem is caused primarily by jusText failing to eliminate leads of related and recommended articles and content coming from index pages containing only article headlines and leads. Leads and headlines of the set of current articles advertised on every article body page during the limited time span of the crawl were thus strongly overrepresented in the corpus.

D. JusText + Onion

JusText [3] was complemented with a post processing tool, called Onion (ONe Instance ONly), which is for removing near-duplicate paragraphs from the corpus. It generates a hash code for each sentence (n-gram of words), and only the first occurrence in the corpus is kept, others are dropped. It can be parametrized to drop whole documents or paragraphs containing duplicated parts. This method effectively decreases the ratio of duplicated content in the corpus, but it often decreases the coherence of the individual texts: they will not be continuous text any more: some parts may be missing from them.

Whether this is a problem or not depends on the aim of the corpus to be gathered. If the goal is just to have a huge collection of sentences, then the available algorithms may perform well enough, the best choice being most probably the jusText+Onion combo. But if it is considered a problem that the title and the lead of an article might be missing while it is attached to just another recent article, i.e. if the coherence of the text is important, then a new approach seems to be needed.

E. CleanEval

CleanEval [4] was a boilerplate remover competition held in 2007. The gold standard corpus used at that competition with a test set of 684 documents is available. The performance of new algorithms on this corpus can be evaluated using an improved evaluation script created by Evert [5]: it calculates precision, recall, F-score, true and false positives and negatives, etc. for the output of a given algorithm. This makes comparison to previously published tools possible.

The documents in the CleanEval corpus were prepared from English and Chinese web pages, which were selected at random: Google results for the following words were retrieved: *picture, extents, raised, events*. Annotators were asked to remove the boilerplate, and to identify the structure of the article (title, paragraphs, lists: using the h, p, l tags). This manually cleaned-up corpus is used as gold standard. The evaluation is based on Levenshtein edit distance [6], adapted by substituting 'token' for 'character'. The calculated edit distance between each pair of cleaned files is divided by the file length: i.e. the percentage of all tokens from either of the two files that cannot be matched with a token in the other file.

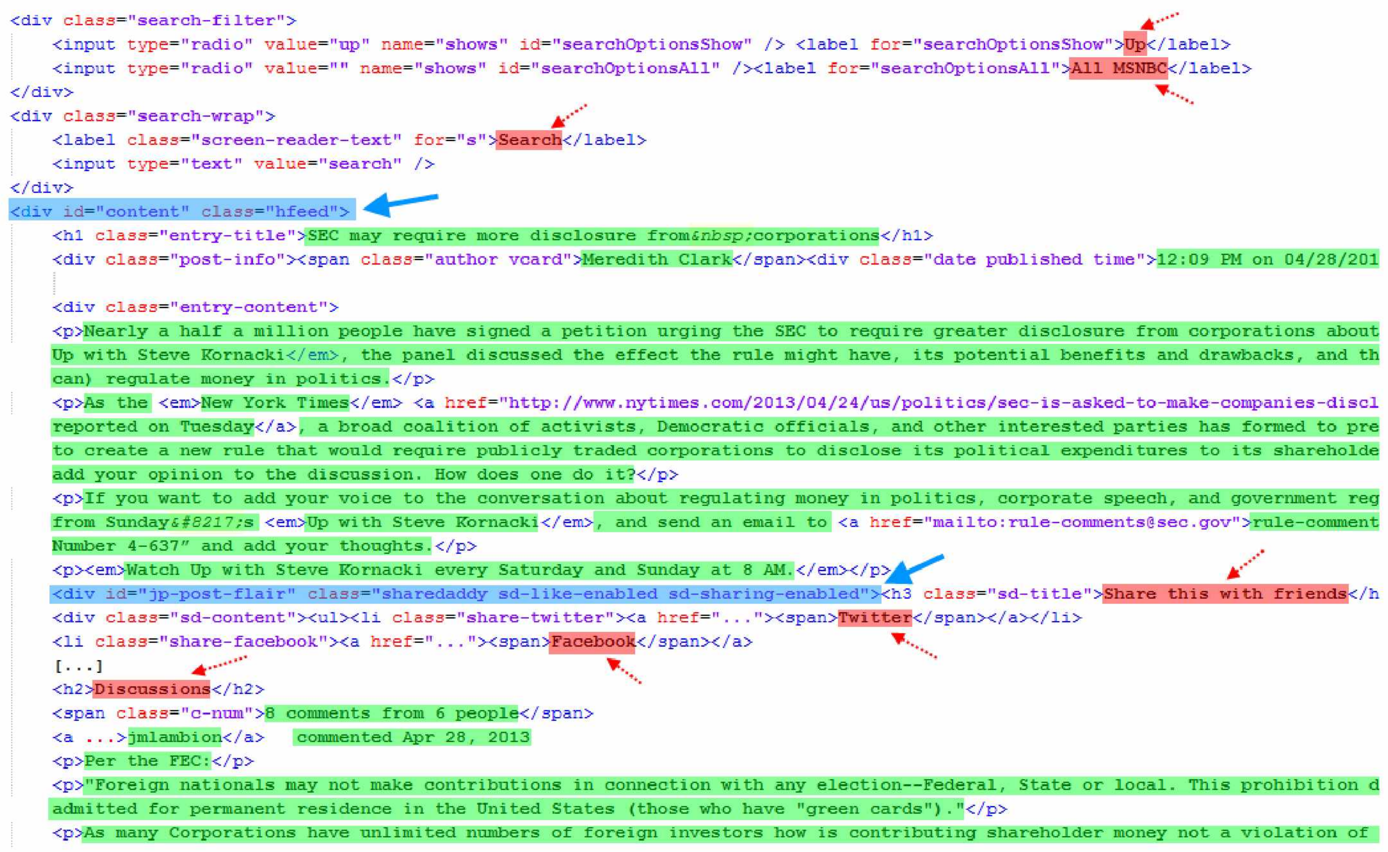
III. THE GOLDMINER ALGORITHM

The problem of boilerplate removal from web pages generated by portal engines can be solved more efficiently if we step up to a level higher than that of individual web pages. As our first attempts at defining a good general procedure for identifying unwanted parts of pages were less successful than expected, we decided to take an optimistic stance and look for what is good instead of what is bad.

We based our approach on the following observations:

1) The relevant part of the HTML content is usually a contiguous stretch (see the BTE approach).
2) Within a web domain/subdomain, the internal structure (the HTML code) of dynamically generated pages generally contains common patterns that can help us identify relevant content.

The algorithm takes a sample of the pages of the domain/subdomain and tries to locate the common patterns in the HTML code within the sample that identify the beginning and the end of valuable content. For example, news portals typically advertise recent and related articles by displaying their headlines and leads next to the actual article. Although this usually seems to be relevant content to jusText, it is in fact just boilerplate content, like menus or advertisements, which has little or nothing to do with the actual article. Not filtering them out results in thousands of duplicates in the corpus.



```
<div class="search-filter">
    <input type="radio" value="up" name="shows" id="searchOptionsShow" /> <label for="searchOptionsShow">Up</label>
    <input type="radio" value="" name="shows" id="searchOptionsAll" /><label for="searchOptionsAll">All MSNBC</label>
</div>
<div class="search-wrap">
    <label class="screen-reader-text" for="s">Search</label>
    <input type="text" value="search" />
</div>
<div id="content" class="hfeed">
    <h1 class="entry-title">SEC may require more disclosure from&nbsp;corporations</h1>
    <div class="post-info"><span class="author vcard">Meredith Clark</span><div class="date published time">12:09 PM on 04/28/201

    <div class="entry-content">
    <p>Nearly a half a million people have signed a petition urging the SEC to require greater disclosure from corporations about
    Up with Steve Kornacki</em>, the panel discussed the effect the rule might have, its potential benefits and drawbacks, and th
    can) regulate money in politics.</p>
    <p>As the <em>New York Times</em> <a href="http://www.nytimes.com/2013/04/24/us/politics/sec-is-asked-to-make-companies-discl
    reported on Tuesday</a>, a broad coalition of activists, Democratic officials, and other interested parties has formed to pre
    to create a new rule that would require publicly traded corporations to disclose its political expenditures to its shareholde
    add your opinion to the discussion. How does one do it?</p>
    <p>If you want to add your voice to the conversation about regulating money in politics, corporate speech, and government reg
    from Sunday&#8217;s <em>Up with Steve Kornacki</em>, and send an email to <a href="mailto:rule-comments@sec.gov">rule-comment
    Number 4-637" and add your thoughts.</p>
    <p><em>Watch Up with Steve Kornacki every Saturday and Sunday at 8 AM.</em></p>
    <div id="jp-post-flair" class="sharedaddy sd-like-enabled sd-sharing-enabled"><h3 class="sd-title">Share this with friends</h
    <div class="sd-content"><ul><li class="share-twitter"><a href="..."><span>Twitter</span></a></li>
    <li class="share-facebook"><a href="..."><span>Facebook</span></a>
    [...]
    <h2>Discussions</h2>
    <span class="c-num">8 comments from 6 people</span>
    <a ...>jmlambion</a>   commented Apr 28, 2013
    <p>Per the FEC:</p>
    <p>"Foreign nationals may not make contributions in connection with any election--Federal, State or local. This prohibition d
    admitted for permanent residence in the United States (those who have "green cards")."</p>
    <p>As many Corporations have unlimited numbers of foreign investors how is contributing shareholder money not a violation of
```

Fig. 1. An example HTML content with unique and not unique paragraphs

Although, as we have seen, post-crawl de-duplication tools, like Onion, can remedy this situation by removing duplicate content, nothing guarantees, however, either that the only remaining instance of the duplicate content is the one that is at the right place or that all duplicates should be removed.

The algorithm learns the HTML tags identifying the beginning and the end of the article for each web domain/subdomain, and only content within this stretch of the page is kept. In addition, since it may still be the case that the body of the article is interrupted with advertisements or other boilerplate content at several points, it is submitted for further processing to the jusText boilerplate removal algorithm. An advantage of this solution is that text from pages with no article content (thematic index pages, tag clouds, search page results, etc.) will not be added to the corpus since the domain-specific HTML tag pattern is not present on them. The algorithm automatically discards the contents of these pages. However, all pages are, of course, still used as a source of URLs for the crawl.

A. A Detailed Description of the Algorithm

The first phase of the crawl of a domain is taking a sample, which is used to identify the domain-specific HTML tag pattern. The algorithm downloads a sample of some 100 pages, applying jusText categorization to each page, which breaks content into paragraphs and evaluates them. Repetitions of individual extracted paragraphs (identified as 'good' by jusText) over different pages in the sample are identified by the GoldMiner algorithm, and these paragraphs are reclassified as bad. Unique paragraphs remain classified as 'good'. Next, it finds the nearest common parent HTML tag of the good paragraphs in the DOM hierarchy on each page. At the end of the learning phase, the most frequent common good parent tag is identified as the winner.

We do not usually get optimal results, however, if the closing tag pair of this parent tag is simply chosen as the tag marking the end of the article. The span enclosed by the parent tag pair may contain bad paragraphs, too. In this case, the algorithm would not find the optimal cutting points. Therefore, it performs another search for the optimal starting and endpoint within the content of the previously selected tag, which may be a series of tags. With the selection of the cutting points, the learning phase for the domain is finished. As the URL domain is crawled afterwards, only the content between the domain-specific beginning and endpoint tag patterns is passed to the jusText boilerplate removal algorithm. Of course, pages used during the learning phase are also handled this way.

During the learning phase, GoldMiner uses only pages where the length of the extracted paragraphs reaches a threshold. Without using a threshold, it failed to learn the optimal cutting points on some domains where thematic opening pages are more frequent than pages containing articles.

B. Illustration of the GoldMiner Algorithm

We present an example in Figure 1 to illustrate the algorithm.

In the learning phase, for every paragraph that was classified as 'good' by JusText, we check if it is unique or not among all pages downloaded from the same domain during the first phase of the crawl. Not unique paragraphs are reclassified as 'bad'. In this example, unique paragraphs are colored green, while those classified either by jusText as boilerplate or those occurring on other pages as well are colored red and marked by small red dotted arrows.

GoldMiner stores html patterns preceding and following green paragraphs. The fragment preceding the green span in the example is:

```
<label class="screen-reader-text"
     for="s">Search</label>
<input type="text" value="search"
     /></div>
<div id="content" class="hfeed">
```

The one following it is:

```
</em></p>
<div id="jp-post-flair"
   class="sharedaddy sd-like-enabled
   sd-sharing-enabled">
```

When the algorithm processed enough pages, it evaluates the stored patterns: it selects the most frequent uniquely identifiable pattern preceding and following the article body. In this example, the best pattern of enclosing tags is highlighted in blue and marked by bigger solid arrows. The configuration information learned for the given subdomain contains these html patterns. The html content of every page is trimmed using these patterns, only the content between the tags matching the patterns will be processed. Thus the otherwise unique content of comments (it also has green color on the picture as it is deemed 'good' by jusText) will be dropped from this page, it will not be considered part of the article.

IV. Evaluation

A. Results and Problems on the CleanEval Corpus

JusText and GoldMiner, with and without Onion post-filtering were tested on the CleanEval test set. As can be seen in Table II, Onion post-filtering increases precision while decreasing recall, which results in net reduction of the balanced F-score.

GoldMiner tries to learn the structure of pages characteristic of each (sub)domain, and applies jusText only to the part of the page that is expected to contain a relevant stretch of text. When comparing the results of GoldMiner with jusText on the CleanEval corpus, we do not get consistent improvement. This is not surprising, though, since this corpus does not contain more than just 3 to 4 pages from each domain, thus GoldMiner has no chance to learn anything relevant about the structure of the pages.

TABLE II
Results on CleanEval

	F-score	Precision	Recall
justText	93.61%	95.29%	91.99%
justText+Onion	93.24%	95.51%	91.08%
GoldMiner	93.40%	95.32%	91.55%
GoldMiner+Onion	93.08%	95.49%	90.78%
BoilerPipe	83.49%	95.15%	74.38%
BTE	91.09%	90.50%	91.68%

TABLE III
Results on CleanPortalEval

	F-score	Precision	Recall
justText	87.26%	78.82%	97.72%
justText+Onion	91.16%	86.48%	96.38%
GoldMiner	98.32%	98.50%	98.15%
GoldMiner+Onion	97.77%	98.48%	97.07%
BoilerPipe	90.68%	92.91%	88.56%
BTE	81.63%	71.20%	95.64%

It is worth mentioning that the corpus contains many torso articles after post-filtering with Onion: Onion often deletes paragraphs from the middle of the text. This often occurs with stereotypical sentences that occur many times in the corpus, like *Good Morning!* etc., and the text is fragmented without them. For example 127.txt in the CleanEval gold standard test set has this text:

```
<h>An open letter to KPLU
<p>To whom it may concern,
<p>Your radio feature by Kirsten
   Kendrick...
```

JusText keeps these paragraphs, but after post-filtering with Onion it looks like this:

```
<h>An open letter to KPLU
<p>Your radio feature by Kirsten
   Kendrick
```

The salutation, *"To whom it may concern,"* is missing. If we want to build a coherent text, not just a collection of independent sentences, the post-filtering performed by Onion may yield suboptimal results.

Moreover, the CleanEval gold standard sometimes does contain boilerplate (e.g. in 634.txt, the last <p> item) or broken words (e.g. 100-102.txt).[1] This, and the wish to demonstrate the power of GoldMiner inspired us to create a new evaluation set: CleanPortalEval, which contains more homogeneous sets of pages.

[1]Serge Sharoff's reaction (p. c.) to calling his attention to this fact: "Nobody is perfect."

TABLE IV
RESULTS ON 2 000 PAGES FROM VARIOUS NEWS PORTALS

Domain	Algorithm	Sentences	Uniq. snt.	%	Characters	Chr. in uniq.	%
origo.hu	BTE	60 682	33 269	54%	12 016 560	7 499 307	62%
	jusText	58 670	30 168	51%	8 425 059	4 901 528	58%
	GMiner	22 475	21 242	94%	3 076 288	3 051 376	99%
nol.hu	BTE	154 547	107 573	69%	24 292 755	13 544 130	55%
	jusText	186 727	128 782	68%	14 167 718	11 665 284	82%
	GMiner	162 674	123 716	76%	12 326 113	11 078 914	89%
index.hu	BTE	51 713	26 176	50%	5 756 176	4 061 697	70%
	jusText	40 970	29 223	71%	4 371 693	3 441 337	78%
	GMiner	13 062	11 887	91%	1 533 957	1 489 131	97%

V. CleanPortalEval

The wish to demonstrate that the approach implemented in GoldMiner is superior to a post-filtering approach for the task of extracting whole articles with minimally compromising the integrity of the texts prompted us to create a new gold standard document set. It contains several pages from the same domain (70 pages from 4 domains), thus it can be used to test the ability of various algorithms to clean pages generated by portal engines. Annotation in this gold standard corpus is similar to that of CleanEval: the output text is annotated using p, h, and l tags by human annotators. The CleanEval evaluation script can be applied to this test set without any changes (the corpus can be downloaded from `https://github.com/ppke-nlpg/CleanPortalEval`). The algorithms were tested on this document set, which yielded the following results, shown in Table III.

Note that, when testing on an appropriate test set that contains enough pages with similar structure, GoldMiner clearly outperforms its rivals both in terms of precision and recall. Applying Onion post-filtering to the GoldMiner output decreases not only recall but also precision in this case (test results can be downloaded from `https://github.com/ppke-nlpg/boilerplateResults`).

VI. Results on Some Portals

Table IV shows the results of the GoldMiner algorithm compared with that of BTE and jusText on three Hungarian news portals: origo.hu, index.hu, nol.hu. The sample corpora quoted in Table IV were generated crawling just the first 2 000 pages from the domains above. Using GoldMiner, the ratio of duplicates in the corpus was reduced considerably compared to what other algorithms produced.

The results clearly show that the algorithm effectively reduces unnecessary duplication in these crawled corpora. Having not revised these pages manually, however, we have no estimate of how the different algorithms perform in terms of the amount/ratio of lost relevant content for these domains.

VII. Conclusion

In this paper, a new boilerplate removal algorithm, GoldMiner, was presented, which can eliminate boilerplate content from dynamically generated web pages in a more efficient way than similar available tools: it identifies recurring HTML tag patterns around relevant content characteristic of web pages coming from a given domain/subdomain. The algorithm preserves textual coherence better than the usual post-filtering de-duplication approach.

A new test document set was created to demonstrate its performance: previous gold standard corpora did not contain enough pages from the same domain for the approach to be applicable. The new gold standard set is called CleanPortalEval and it is open to the public.

Acknowledgments

This research was partially supported by the project grants TÁMOP–4.2.1./B–11/2-KMR-2011-0002 and TÁMOP–4.2.2./B–10/1-2010-0014.

References

[1] 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.

[2] C. Kohlschütter, P. Fankhauser, and W. Nejdl, "Boilerplate detection using shallow text features," in *Proceedings of the third ACM international conference on Web search and data mining*, ser. WSDM '10. New York, NY, USA: ACM, 2010, pp. 441–450. [Online]. Available: http://doi.acm.org/10.1145/1718487.1718542

[3] J. Pomikálek, "Removing boilerplate and duplicate content from web corpora [online]," Ph.D. dissertation, Masarykova univerzita, Fakulta informatiky, 2011.

[4] M. Baroni, F. Chantree, A. Kilgarriff, and S. Sharoff, "Cleaneval: A competition for cleaning web pages," in *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*, B. M. Nicoletta Calzolari, Khalid Choukri and D. Tapias, Eds. Marrakech, Morocco: European Language Resources Association (ELRA), 2008.

[5] S. Evert, "A lightweight and efficient tool for cleaning web pages," in *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*, B. M. Nicoletta Calzolari, Khalid Choukri and D. Tapias, Eds. Marrakech, Morocco: European Language Resources Association (ELRA), 2008.

[6] V. Levenshtein, "Binary codes capable of correcting deletions, insertions, and reversals," *Cybernetics and Control Theory*, vol. 10, no. 8, pp. 707–710, 1966, original in *Doklady Akademii Nauk SSSR* 163(4): 845–848 (1965).