

# Web Corpus Cleaning using Content and Structure

Katja Hofmann<sup>1</sup>, Wouter Weerkamp<sup>1</sup>  
ISLA, Universiteit van Amsterdam

## Abstract

This paper describes experiments on cleaning web corpora. While previously described approaches focus mainly on the visual representation of web pages, we evaluate approaches that rely on content and structure. We have evaluated a heuristics-based approach, as well as approaches based on decision trees, a genetic algorithm and language models. The best performance was achieved using the heuristics-based approach.

**Keywords :** web corpus, content extraction, heuristics, genetic algorithm, language models, decision tree.

## 1. Introduction

Research at the Information and Language Processing Systems (ILPS) group at the University of Amsterdam focuses on natural language processing and information retrieval. Current projects include information extraction, expert finding, opinion mining, blog search, and question-answering. All these research activities require the use of sizable text corpora. A resource that is becoming more and more important is the web: it offers a virtually unlimited amount of documents and therefore huge opportunities and also challenges to researchers. In addition ILPS participates in many different tasks within competitions like TREC<sup>1</sup> and CLEF<sup>2</sup>. Most of these tasks are moving towards the use of web corpora as well. Web corpora we are currently dealing with include the TREC Blog06 corpus and the TREC web corpus, among others.

A web corpus consists of web pages, documents that are usually marked up using HTML. Besides containing interesting and relevant information, these HTML files also contain navigational structure (i.e. menus), headers (logos, breadcrumbs), footers (copyright notice, dates), and advertisement. One of the main problems when creating a web corpus is that these unwanted pieces of text generate noise, making processing much harder.

These problems call for a well-designed content extraction approach. In entering the CleanEval<sup>3</sup> competition we make a first structured attempt in developing a content ex-

---

<sup>1</sup> ISLA, Universiteit van Amsterdam, {khofmann,weerkamp}@science.uva.nl

<sup>1</sup> <http://trec.nist.gov/>

<sup>2</sup> <http://www.clef-campaign.org/>

<sup>3</sup> <http://cleaneval.sigwac.org.uk/>

traction system that can assist us in all previously mentioned tasks. CleanEval is the first attempt to a joint effort in web page cleaning. Participants are required to run their respective systems on a document set and their output is evaluated against manually cleaned documents. Expected output consists of the textual content with a limited amount of markup distinguishing between headings, list items, and paragraphs. Prior to the competition a small test document set was made available and could be used for training and initial evaluation.

In this paper we discuss the approaches we took for our ILPS Content Extraction (ICE) system and our initial evaluation results. Section 2. will give some information on approaches that have been tried before, after which we will discuss our own approaches in Section 3. Section 4. discusses the results of these approaches and Section 5. lists our conclusions and ideas for future research.

## 2. Related Work

The cleaning of web pages has rarely been described in the literature so far and performance of the developed systems has not yet been compared in a systematic way. In most cases, content extraction from web pages is considered a preprocessing phase, and its output is used as input for a next phase. Instead of being evaluated directly, evaluation of the preprocessing phase is based on performance of the complete system.

Song et al. (Song *et al.* 2004) use machine learning to identify important block level elements in web pages from visual (or spatial) features. In the described system, window-based spatial features (placement and sizes of blocks relative to the browser window) appear to be good features in identifying important blocks, while adding content features (e.g. length, number of images and number of hyperlinks) improves performance only marginally. The approach is compared to human performance and achieves competitive results. In (Cai *et al.* 2003) this approach is evaluated on an information retrieval task; the authors use a pseudo-relevance feedback task to compare the performance when using visual representations to the performance when using the complete document. An overall 27% improvement is measured when using a cleaned corpus (i.e. containing important blocks).

Yi & Liu (Yi & Liu 2003) discuss a tree-based approach that combines features based on HTML tree structure, content, and visual representation. The authors construct a so-called compressed structure tree; elements in the tree of each document are combined (compressed) if their child elements share the same tag names, attributes, and attribute values. Based on the number of different presentation styles for an element, the weight (importance) of this element is determined. The resulting weights are then used in follow-up tasks (e.g. classification).

While this approach performed well in the follow-up classification task, it depends on the availability of a relatively large number of documents from a limited number of sources. In their paper, Yi & Liu used documents from 5 distinct sources. The documents from each source had very similar structure which made the use of the compressed structure

tree a viable option. In the task at hand we expect documents from a large number of very different sources and constructing a compressed structure tree therefore seems unrealistic. Furthermore, this approach depends on the availability of presentational information inside the documents, which may or may not be available for the provided documents.

### 3. Approach

Extraction of informative content from web pages can be based on a) visual representation b) HTML tree structure, c) textual content, or any combination of the above. As approaches based on visual representation have been followed most extensively in the past, we are focusing on the use of content and, to some extent, structure. Extraction of informative text based on content features could take content length (in either words or characters), capitalization or punctuation into account. More advanced methods could be based on information retrieval approaches like *tf.idf* to determine the topic of a document and extract relevant information based on this topic. Features based on document structure could include information on parent or child elements within the HTML tree, or the order of elements.

We are submitting three runs to CleanEval: one baseline, one heuristics-based run, and one run based on decision trees (DT). Additionally, we describe experiments using language models (LM) and genetic algorithms (GA). LM has shown promising results but is highly dependent on the quality of the available content. The GA-based approach failed to improve the baseline results, but is included here as a possible basis for future research.

#### 3.1. Baseline

Our baseline run uses an HTML parser<sup>4</sup> to extract all block-level elements (*p*, *div*, *li*, *h1..6*, or plain text) from the original HTML. During this step, we attempt to convert all text to UTF-8 encoding based on the meta-tag information, if included in the HTML page. In addition, we replace two or more subsequent HTML line breaks (`<br>` tags) with paragraph boundaries, as we have observed that the two are used interchangeably to indicate paragraphs.

Our baseline assumes all block level elements to constitute paragraphs and outputs those using the markup `<p>`. We do not remove any text during this run. Headings (`<h>`) or list items `<li>` are not used.

#### 3.2. Heuristics-Based Approach

In our heuristics-based approach we use the Jericho HTML Parser<sup>5</sup> to remove all HTML tags and keep only the textual content of either the *body* element or the top level element (in case of malformed HTML code). After splitting the remaining text into separate lines we use word and character length heuristics to select relevant text. Figure 1 lists

---

<sup>4</sup> We used the perl module `HTML::TreeBuilder` available from CPAN (<http://cpan.org>)

<sup>5</sup> available from <http://jerichohtml.sourceforge.net>

the algorithm that is used in this step.

```

for each L in lines
  if character length L > 3
    split L into words
    if number of words in L > 10 and average character length of words in L > 3
      if possible_title is set and line counter < 3
        add possible_title as header
        unset line counter, possible_title
      endif
      add L as paragraph
    else
      set possible_title to L
      reset line counter
    endif

  else if possible_title is set
    line counter ++
  endif
end for

```

*Figure 1. Pseudo-code for our heuristics-based web page cleaning approach*

This simple approach identifies paragraphs and headings and does not take list items into account. To identify candidate headings we use length in words. If the candidate heading is followed by a longer piece of text within a certain window we mark the candidate as a heading and the subsequent text as paragraph element.

More elaborate heuristics have been evaluated, but were not submitted for run-time performance reasons. These heuristics try to capture the notion of list items; we experimented with heuristics that decide for each element in the HTML tree whether it is a paragraph, heading, list item or whether it should be discarded. The list item identification is done based on the text's parent tag (`<li>` or not), length in words (decreasing list item probability per word), capitalization and the likelihood of the previous and next element being list items. For the heading identification we use similar heuristics: length in words, parent tag (`<h1..6>` or not) and whether or not the next element contains more than 10 words. Both the list item and the heading should not be or contain a hyperlink. For the paragraph decision we look at the ratio of punctuation to total number of words and the ratio of conjunction words to total number of words. Based on these three pseudo-probabilities we can decide on which markup to give to the element. If the element is considered a paragraph its score should be above a certain threshold to be included in the final output. Setting this threshold is one of the issues in this approach.

The more elaborate heuristics give us the possibility to identify elements as a list item, but this also makes this approach more error prone. To eliminate the human error in determining the correct settings for the heuristics and to improve generalizability we explore the possibilities of decision trees in the next section.

```

wordcount <= 15 :
| parent_tag = h1: h (35.0/7.2)
| parent_tag = h5: n (25.0/1.3)
| parent_tag = h6: n (0.0)
| parent_tag = div:
| | wordcount <= 2 :
| | | cap_ratio <= 1.25 :
| | | | punct_ratio > 0.25 : n (55.0/8.3)
| | | | punct_ratio <= 0.25 :
| | | | | cap_ratio <= 0.25 :
| | | | | | wordcount <= 1 : h (67.0/1.4)
| | | | | | wordcount > 1 : n (2.0/1.0)
| | | | | | cap_ratio > 0.25 :
| | | | | | | cap_ratio <= 0.75 : n (38.0/3.8)
| | | | | | | cap_ratio > 0.75 : h (222.0/74.4)
| | | | | cap_ratio > 1.25 :
| | | | | | wordcount <= 1 : n (3.0/2.1)
| | | | | | wordcount > 1 : p (11.0/2.5)
| | | wordcount > 2 :
| | | | cap_ratio > 0.218182 : n (799.0/92.8)
| | | | cap_ratio <= 0.218182 :
| | | | | punct_ratio <= 0.0333333 : n (14.0/5.8)
[...]
```

Figure 2. Sample of the learned decision tree classifier used in our experiments. Nodes can be read as if-then rules, e.g. if wordcount  $\leq 15$  and parent\_tag = h1 then classify item as h. The numbers in parenthesis specify the number of items classified according to a particular path and, if applicable, the number of misclassified items. Fractions can occur due to probabilistic weighting of training instances.

### 3.3. Decision Trees

Decision trees (Quinlan 1993) are frequently used for statistical classification. The trees' nodes define feature values based on which items are classified. The trees can be induced (learned) automatically from a set of labeled examples.

In our experiments we attempt to classify block level elements into one of the four classes  $h$  (heading),  $l$  (list item),  $p$  (paragraph), and  $n$  (not included).

The training data is derived from the CleanEval test set. We match block level elements with the manually cleaned files. If the element is found, it is marked according to its manually assigned label. Otherwise, the element is labeled as  $n$ . In this way, 5866 labeled training instances are obtained, with 3467 labeled  $n$ , 1311 labeled  $p$ , 663 labeled  $h$ , and 425 labeled  $l$ . Our method of extracting instance labels assumes that manually cleaned files align with block level elements. When this is not the case, errors may be introduced, biasing the training data set towards not including elements in the cleaned output.

The choice of a feature set is based on our experiments using heuristics: the number of words a piece of text consists of (*wordcount*), the HTML tag it is enclosed in (*parent\_tag*), the ratio of words to capitalized words (*cap\_ratio*), the ratio of words to punctuation (*punct\_ratio*), and the ratio of words to stop words (*stop\_ratio*).

To run our experiments, we use the C4.5 software package<sup>6</sup>. We first evaluate our approach using ten-fold cross-validation which resulted in an average error rate of 22.57%. Following this initial evaluation, we train a decision tree classifier using all 5866 labeled instances extracted from the CleanEval data. A sample of the trained classifier is shown in figure 2.

### 3.4. Language Models

Although not submitted as an official run, we experimented with using language models (LMs) to determine which content should be extracted. LMs are frequently used in the field of Information Retrieval (IR), (Ponte & Croft 1998), (Song & Croft 1999) and (Hiemstra 2001). The rationale behind using LMs in IR is that the user query could have been generated by one (or more) documents in the document set. An LM approach to IR tries to determine the probability of the query being generated by each document and returns the documents with the highest probabilities. To do this LMs of both the query and the document are constructed and the difference between them is calculated.

We can consider extracting content from web pages an IR problem: we must determine which elements in the document are relevant to the user. The main problem in this approach is the lack of a query and we therefore do not know upfront what content would be considered relevant. The challenge is to build a query from the information available on the web page. In various papers the importance of titles, keywords and headings of web pages to IR is mentioned (Hu *et al.* 2005): since headings and titles in web pages are more important when determining the relevance of a document regarding a query, we might conclude that the headings and titles are stronger related to a possible query than other elements on a web page. This assumption gives us the possibility to construct a query by combining all headings, titles and keywords from a page.

After constructing an LM for the 'query' we remove inline elements from the HTML page construct an LM for each of the leaf elements. All LMs are constructed using absolute discount smoothing (Zhai & Lafferty 2004), although more sophisticated methods could be used (e.g. Dirichlet smoothing). We calculate the Kullback-Leibler divergence between every element LM and the query LM and consider elements with an absolute KL-divergence smaller than a threshold to be relevant. As with the paragraph detection in Section 3.2. setting this threshold is difficult and should be investigated further. We did not submit an official run using the language model approach, but results on the test set are included in Section 4.

---

<sup>6</sup> Available from <http://www.rulequest.com/Personal/c4.5r8.tar.gz>

### 3.5. Genetic Algorithm

In addition to the submitted runs we conducted experiments using a genetic algorithm (GA). GAs form an example of feedback learning that imitates biological evolution. Starting with a random population of string-encoded candidate solutions to a given problem, the population evolves based on selection, crossover, and mutation. In each generation, individuals are assigned a fitness score indicating their performance on the given problem. The fittest individuals are selected and new individuals are created via crossover of the selected individuals. Randomly, mutation can be applied as well. Performance of GA depends on the ability to find a suitable encoding for candidate solutions, as well as giving appropriate feedback via the assigned fitness score.

The goal of our experiments is to use GA to evolve a regular expression that extracts content from web pages. Our experiments are based on an existing GA implementation<sup>7</sup>. We encode candidate solutions as Trees, which can be translated into regular expressions. Custom initialization, crossover and mutation strategies are developed to ensure that each generated individual can be translated into a valid regular expression.

We experiment with a subset of perl regular expression syntax: capturing nodes, non-capturing nodes, OR-nodes, and *literals*. Literals are implemented as leaf nodes and contain any one of a fixed set of manually specified items (`\w*?`, `<.*?>`, ...).

Candidate solutions are evaluated as follows. The candidate is translated into a regular expression which is repeatedly matched against the original HTML document until no further matches are found. When a match is found, all content of capturing groups is kept, remaining text is discarded. Initially, the extracted text was evaluated using the scoring script provided by CleanEval. However, this script is too slow to train GA effectively.

As a simple scoring function we use a bag-of-words comparison between the manually cleaned file and the output of the candidate solution. We award 2 points for each word from the candidate solution that occurs in the manually cleaned file, and deduct 1 point for each excess word that is not part of the manually cleaned file. We use this weighting scheme based on the assumption that high recall is desired, i.e. it is more important that all content text is kept than it is to discard all noise.

Due to time constraints we are only able to run a small number of experiments using this setup. In addition, the evaluation of random regular expressions poses problems in that run-time can exceed reasonable amounts of time. We solve this problem using a time-out that interrupts evaluation when a threshold is exceeded. The best solution simply returns all words contained in the original HTML page. Therefore this approach does not improve over the baseline approach.

---

<sup>7</sup> AI::Genetic, available from CPAN (<http://cpan.org>)

|        | text-alignment |      |             | text-only |      |             | combined score |      |             |
|--------|----------------|------|-------------|-----------|------|-------------|----------------|------|-------------|
|        | min            | max  | average     | min       | max  | average     | min            | max  | average     |
| ice_bl | 17.5           | 81.5 | 50.7        | 0.0       | 99.4 | 73.7        | 16.7           | 90.4 | 62.2        |
| ice_hr | 16.7           | 94.0 | <b>58.7</b> | 0.0       | 99.8 | <b>78.3</b> | 8.3            | 96.3 | <b>68.5</b> |
| ice_dt | 0.0            | 86.6 | 47.6        | 0.0       | 98.6 | 57.3        | 0.0            | 92.6 | 52.4        |
| ice_lm | 0.1            | 89.4 | 51.4        | 0.0       | 98.9 | 69.9        | 0.0            | 93.9 | 60.6        |

Table 1. Minimum, maximum and average of text-alignment, text and combined scores of the evaluated methods on the test data set provided through CleanEval. Best average scores are highlighted in bold. ice\_lm was not included as a separate evaluation run and is listed for comparison only.

## 4. Evaluation

All approaches are tested on the CleanEval test set in order to estimate their performance. Performance is measured using an evaluation script through the CleanEval competition. The script calculates text-only, text-alignment and combined scores. The text-only scores are highest when all noise is discarded and all informative content is preserved. Text-alignment scores also take markup as headings, list items and paragraphs into account. Table 1 gives an overview of the performance of the systems submitted to CleanEval, as well as our best language model based cleaner.

Our baseline (*ice\_bl*) shows relatively strong and consistent performance with an average combined score of 62.2%. Text-only scores are consistently higher than text-alignment scores, because we tagged all elements as paragraphs, thereby ignoring headings and list items. This approach performs best on 32% of the test files.

The heuristics-based approach (*ice\_hr*) shows the best performance on the CleanEval test set. It outperforms the baseline with a 15.8% relative improvement of text-alignment scores, a 6.2% relative improvement of text-only scores, and a 10.1% relative improvement of combined scores. Furthermore, this approach performed best on 36% of the test files. Potential problems with this approach are due to overfitting: only a small number of web pages can be inspected manually, which means that the algorithm may be tuned to the test documents but may not perform well on an unseen document set.

The more elaborate decision tree approach performs low on average with scores below the baseline. However, the approach performs best on 20% of the test files. Error analysis shows that low scores mainly resulted from discarding too many elements. Thus, this approach shows some promise but more experiments are necessary. We are planning to evaluate this approach using a cleaner training set, more fine-grained features, as well as different weighting schemes.

In addition to the three runs submitted to CleanEval, we include a run based on language models (*ice\_lm*) in our evaluation. On average, performance is comparable with the baseline, with slightly better text-alignment scores (since it does include headings) and slightly lower text-only scores. However, this algorithm performs best on only 11% of the test files. As the LM for each file is based on information available in headings and keyword annotations, this approach does not perform well when little or no

such information is available, while it performs well when high-quality information is available. Therefore, LM approaches may be most promising in combination with other approaches.

## 5. Conclusion

Our participation in CleanEval focused on exploring the possibilities of cleaning a web corpus using content and structure, and thus ignoring representational features. The evaluation results show that for the relatively small test set a heuristics-based approach works best; tuning of the heuristics is still feasible due to the small set. Evaluation on a larger document set will show how generalizable the identified heuristics are. To explore the problem space of the feature set developed for the heuristics more systematically we evaluate a decision tree approach. Initial results are mixed and further experiments are required. The LM approach is interesting from an IR point of view, but is highly dependent on the quality and amount of headings and keywords in each document.

We would like to comment on one particular characteristic of the CleanEval competition. In our experience we do not require identification of list items by the cleaning systems. Overall, we would favor a system that performs very good on text-only over a system that performs reasonably well on both text-only and markup.

Future research aims at improving the decision tree approach by using more training documents and possibly more sophisticated features. Including the KL-divergence as a feature in the decision tree could be a way of combining the decision tree and LM approaches. Improving the stand-alone LM approach would call for better estimation of the query LM (e.g. include text in *bold* tags in the LM), better parameter setting and more sophisticated smoothing methods.

## Acknowledgments

Katja Hofmann is supported by NWO under project number C.2324.0114.

## References

- CAI D., YU S., WEN J.-R. et MA W.-Y. (2003), "Extracting content structure for web pages based on visual representation", in *Proc. 5th Asia Pacific Web Conference*.
- HIEMSTRA D. (2001), *Using Language Models for Information Retrieval*, PhD thesis, University of Twente, Enschede.
- HU Y., XIN G., SONG R., HU G., SHI S., CAO Y. et LI H. (2005), "Title extraction from bodies of HTML documents and its application to web page retrieval", in *SIGIR '05: Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*: ACM Press, New York, NY, USA : 250–257.
- PONTE J. et CROFT W. (1998), "A Language Modeling Approach to Information Retrieval", in *Proceedings of SIGIR 98* : 275–281.
- QUINLAN J. R. (1993), *C4.5: Programs for Machine Learning*, Morgan Kaufmann.
- SONG F. et CROFT W. B. (1999), "A General Language Model for Information Retrieval (poster abstract)", in *Research and Development in Information Retrieval* : 279–280.

- SONG R., LIU H., WEN J.-R. et MA W.-Y. (2004), “Learning block importance models for web pages”, in *WWW '04: Proceedings of the 13th international conference on World Wide Web*: ACM Press, New York, NY, USA : 203–211.
- YI L. et LIU B. (2003), “Web Page Cleaning for Web Mining through Feature Weighting”, in *Proceedings of Eighteenth International Joint Conference on Artificial Intelligence (IJCAI-03)*.
- ZHAI C. et LAFFERTY J. (2004), “A study of smoothing methods for language models applied to information retrieval”, in *ACM Trans. Inf. Syst.*, n° 2, vol. 22.