A lightweight and efficient tool for cleaning Web pages

Stefan Evert

Institute of Cognitive Science, University of Osnabrück
49069 Osnabrück, Germany
stefan.evert@uos.de

Abstract

Originally conceived as a "naive" baseline experiment using traditional n-gram language models as classifiers, the NCLEANER system has turned out to be a fast and lightweight tool for cleaning Web pages with state-of-the-art accuracy (based on results from the CLEANEVAL competition held in 2007). Despite its simplicity, the algorithm achieves a significant improvement over the baseline (i.e. plain, uncleaned text dumps), trading off recall for substantially higher precision. NCLEANER is available as an open-source software package. It is preconfigured for English Web pages, but can adapted to other languages with minimal amounts of manually cleaned training data. Since NCLEANER does not make use of HTML structure, it can also be applied to existing Web corpora that are only available in plain text format, with a minor loss in classfication accuracy.

1. Introduction

The World Wide Web is an amazing, almost inexhaustible and very convenient source of authentic natural language data. Automatically compiled Web corpora have become increasingly popular in recent years and have been used for many different purposes (Kilgarriff and Grefenstette, 2003; Baroni and Bernardini, 2006). In particular, Web corpora offer the NLP community an opportunity to train statistical models on, and mine information from, much larger amounts of text than was previously possible. It is easy and inexpensive to collect billions of words of English, German, French, and many other languages.

NLP researchers sometimes use "large and messy" training corpora in the hope that errors will somehow cancel out in the statistical models (Banko and Brill, 2001). Web pages are even messier than other text sources, though, and interesting linguistic regularities may easily be lost among the countless duplicates, index and directory pages, Web spam, open or disguised advertising, and boilerplate. Therefore, thorough preprocessing and cleaning of Web corpora is crucial in order to obtain reliable frequency data.

Software tools are readily available for many subtasks, especially those that are also relevant to Web search engines: identification of language and encoding, duplicate removal, detection of Web spam and automatically generated pages, HTML-to-text conversion, and possibly also automatic stripping of advertisements.[1] When these steps have been performed, the resulting pages may still contain substantial amounts of linguistically undesirable material, in particular any kind of canned or automatically generated text blocks. Collectively referred to as *boilerplate*, such canned text includes navigation bars, page headers, link lists, disclaimers and copyright statements, as well as the ubiquitous advertisements. Similar to page duplicates, boilerplate may grossly inflate frequency counts for certain terms and expressions (such as *click here*, *Contents* or *Vi@gr@*). Therefore, reliable boilerplate removal is indispensable for the compilation of useful Web corpora.

Commercial search engines (as well as other popular and well-funded application domains such as information retrieval) are usually not concerned about boilerplate, which will not figure prominently among users' search results, especially when ranking techniques are applied. Even in the Web as Corpus community, this task has often been trivialised or ignored. Before the recent CLEANEVAL competition,[2] boilerplate removal was typically performed with simple ad-hoc scripts, if at all.

Team	Text	Seg
Bauer et al. (Osnabrück)	73.5	53.5
Marek, Pecina & Sprousta (Prague)	**84.1**	65.3
Hofmann & Weerkamp (Amsterdam)	83.0	65.5
Chaudhury (India)	80.9	59.5
Conradie (South Africa)	60.2	45.5
Gao & Abou-Assaleh (GenieKnows)	83.4	63.9
Girardi (IRST)	82.5	**65.6**
Saralegi & Leturia (Elhuyar Foundation)	83.4	65.3
Evert (Osnabrück)	*82.9*	*60.3*

Table 1: CLEANEVAL competition results (Text = text cleaning, Seg = paragraph segmentation and classification).

The page cleaning tool presented in this paper started out as a "naive" baseline experiment to find out whether character-level n-gram models would be able to distinguish between clean body text and boilerplate. In particular, no use was made of HTML tags and structure, in stark contrast to previous approaches that rely heavily on tag density and other heuristic measures.[3] The n-gram models worked surprisingly well, achieving a significant improvement over unfiltered text dumps. In the CLEANEVAL competition held in 2007, its text cleaning accuracy came close to those of the best systems (English track, text only, cf. Table 1). Paragraph segmentation and classification was not an objective of the experiment, and as a result the algorithm performed

[1] In the author's experience, however, Google search results often seem to be dominated by advertising and pages containing counterfeit category listings or search results.

[2] See `http://cleaneval.sigwac.org.uk/`

[3] A typical example is the *PotaModule*, a Perl script available from `http://sslmitdev-online.sslmit.unibo.it/wac/post_processing.php`.

considerably worse in this category.[4]

Since the n-gram based approach is also lightweight, fast and portable, it has been repackaged as an open-source standalone tool named NCLEANER. It is distributed in the form of a Perl package and can be downloaded from `http://webascorpus.sf.net/`. While the first release only includes a standard parameter file for English, the statistical algorithm should in principle work equally well for all alphabetic languages, and requires only minimal amounts of training data for each new language (between 30 and 50 manually cleaned Web pages). A particular advantage of NCLEANER is that it can also operate on plain text files, e.g. existing Web corpora for which the original HTML pages are no longer available. It might also be useful for stripping advertisements and other boilerplate from mailing list and newsgroup postings.

2. System architecture

The NCLEANER system is a pipeline of four modules, as illustrated in Figure 1. When the system is applied to existing text dumps, only the last two components are utilised.

The *first* stage of the pipeline performs some normalisation on the original HTML code. Regular expressions are used to delete images, comments and inline JavaScript. In addition, special markers are inserted for headings and list items, and `<br/>` tags are converted to regular paragraph breaks (in order to improve paragraph segmentation and classification).

In the *second* stage, the pre-processed HTML page is converted to plain text using the text-mode browser Lynx.[5] NCLEANER relies on the built-in character encoding recognition of the browser, and obtains a normalised text file in UTF-8 encoding.

In a *third* post-processing step, invalid characters are removed from the text, and some easily recognisable types of boilerplate (e.g. navigation bars or footer blocks where multiple fields are separated by vertical bars "|") are deleted with regular expressions. Itemised and enumerated lists in the Lynx output are detected, as well as paragraph breaks indicated by empty lines. In this way, the text is broken down into segments, which are classified as `<p>` (general paragraph), `<h>` (heading) or `<l>` (list item), following the CLEANEVAL annotation guidelines. The identification of different segment types relies in part on the markers inserted by the first module.

The *fourth* and core component of the NCLEANER algorithm consists of two separate character-level n-gram language models for "clean" and "dirty" text. These models are applied to each identified text segment in turn. If the dirty model calculates a higher probability than the clean model, the text segment is considered to be boilerplate and deleted. Otherwise it is included in the final output.

The research prototype for the CLEANEVAL competition has been implemented in pure Perl (including the n-gram models) and is therefore completely platform-independent. The standalone tool also offers a more efficient implementation mixing C and Perl code, which is automatically activated on supported platforms.

[4] Note that NCLEANER participated into the CLEANEVAL competition under its working title *StupidOS*. The name change was recommended, rather wisely, by an anonymous reviewer.

[5] `http://lynx.browser.org/`

3. Training the language models

In order to distinguish between boilerplate and clean text, the NCLEANER system uses simple character-level n-gram language models (Manning and Schütze, 1999), based on conditional probabilities $\Pr(c_n \mid c_1 \ldots c_{n-1})$, where $c_i \in \Sigma$ are characters in the input alphabet. Since it was originally developed for English text, Σ is restricted to the ASCII range, and non-ASCII characters have to be folded appropriately. Full Unicode support would lead to extreme data sparseness problems. In the current implementation, non-ASCII characters are replaced by tilde symbols (~), but more sophisticated folding or support for selected Unicode character ranges can easily be implemented in the post-processing stage.

To compensate for the small, uniform training corpus and ensure better generalisation, geometric interpolation is used to estimate n-gram probabilities for unseen text:

$$\Pr_{\text{smoothed}}(c_n \mid c_1 \ldots c_{n-1}) = \frac{1-q}{1-q^n} \cdot \Big(\Pr(c_n \mid c_1 \ldots c_{n-1}) + q \cdot \Pr(c_n \mid c_2 \ldots c_{n-1}) + \cdots + q^{n-1} \cdot \Pr(c_n) \Big) \quad (1)$$

with parameter parameter $q \in (0, 1)$. Note that $q \approx 0$ corresponds to an unsmoothed n-gram model without interpolation, whereas $q \approx 1$ gives equal weight to all history sizes. In addition, add-one smoothing is applied to unigram probabilities $\Pr(c_n)$ in order to account for unseen and infrequent characters in the training data.

The training data consist of 158 English Web pages from several topical domains (including "Norway", "computer hardware", "photography" and "Japanese manga and anime"), which were cleaned manually starting from Lynx text dumps. Each page was cleaned independently by two annotators, and conflicts were resolved by a third annotator. In total, the training data amount to more than 300,000 words and almost 2 million characters (after manual cleanup). A detailed description of this gold standard is given by Bauer et al. (2007).

Unfortunately, the gold standard only provides raw text dumps and manually cleaned files, but does not explicitly mark boilerplate in the text dumps. Therefore, standard maximum-likelihood estimation of conditional probabilities was only possible for the "clean" n-gram model. For the "dirty" model, a *differential training* scheme had to be used: n-gram counts for the clean pages were subtracted from the corresponding n-gram counts for the raw text dumps, the differences between the two values approximating the n-gram counts that would be obtained from an explicit boilerplate training corpus.

The best n-gram size n and the optimal value for the interpolation parameter q were determined by minimisation of cross-entropy on the raw text dumps, calculated by 10-fold

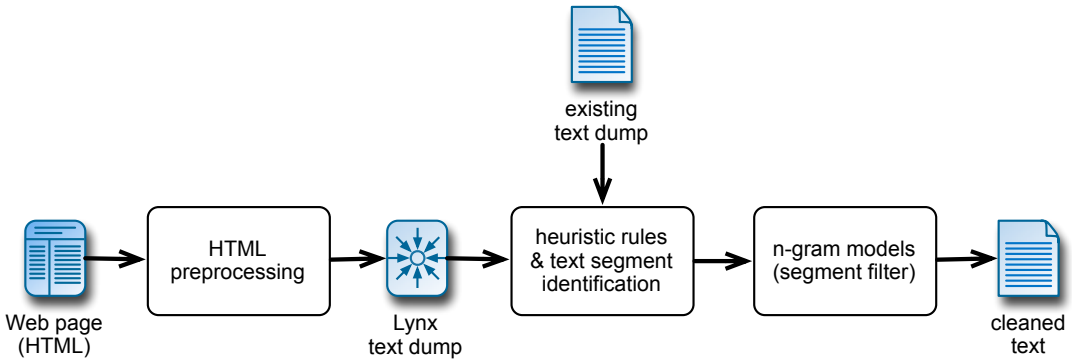

Figure 1: Architecture of the NCLEANER system

cross-validation.[6] Figure 2 plots cross-entropy against the interpolation parameter q for various values of n. These graphs suggest $n = 6$ and $q = 0.6$ (a fairly strong interpolation where unigrams still have a relative weight of 0.03) as a good and robust choice, with a cross-entropy of 2.53 bits. Note that higher-order models (e.g. $n = 8$) require extremely strong interpolation, indicating a low degree of robustness.

Evaluation of the full system revealed that lower-order models achieve slightly better accuracy despite higher cross-entropy values, perhaps because of their better generalisation capabilities. The current implementation uses models with $n = 3$ and $q = 0.5$, but experiments indicate that overall performance is robust across a wide range of parameter values.

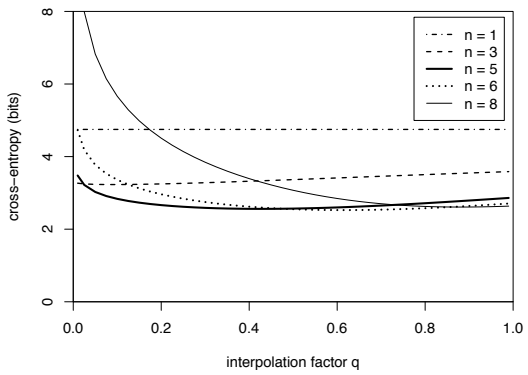

Figure 2: Parameter tuning for the n-gram language models on joint "clean" and "dirty" training data.

A drawback of the n-gram approach is that it is highly language-specific, and separate training data are required for each new language the system is applied to. In order to test whether it might be feasible to build a standard parameter file that works well for multiple languages, *non-lexical* versions of the n-gram models have also been implemented, which map all letters to `a` and all digits to `0`. These models cannot learn to recognise language-specific keywords and have to rely on features such as word length and the presence of special characters (such as punctuation) for their classification.

[6] Note that a straightforward minimisation of cross-entropy on the training data without cross-validation would lead to serious overfitting. The raw text dumps were used because they comprise both the clean and the "virtual" boilerplate training corpus.

4. Internal evaluation

An internal evaluation of the NCLEANER algorithm was carried out by cross-validation on the gold standard of Bauer et al. (2007). In the CLEANEVAL competition, minimum edit distance was used as an evaluation criterion. However, there are two important drawbacks to this approach: (i) the resulting scores are not very intuitive and fail to distinguish between false positives and false negatives (which may have different importance in practical applications); (ii) the minimum edit distance algorithm is computationally expensive, and the Perl implementation provided by the CLEANEVAL organisers was far too slow to be used for extensive experiments.

Therefore, the evaluation of NCLEANER was carried out in terms of word-level *precision* (i.e. the percentage of correct words in the automatically cleaned files) and *recall* (i.e. the percentage of clean words in the gold standard that were preserved by the automatic procedure). As an overall measure, *F-score* (the harmonic mean of precision and recall) was used. Precision and recall were determined from a word-level sequence alignment of the automatically cleaned pages with the corresponding gold standard versions, using a fast heuristic algorithm implemented in the Python `difflib` package.

Table 2 shows precision, recall and F-score for (i) Lynx text dumps as a baseline, (ii) the NCLEANER pre- and post-processing heuristics *without* the language models, (iii) the full NCLEANER system, and (iv) the non-lexical version of the language models. *Micro-averaged* values are percentages of correct words in the entire gold standard (i.e. pooling data from all files); *macro-averaged* values calculate the average and standard deviation of evaluation scores across the individual files. Note that macro-averaged scores are considerably lower than micro-averaged ones, which is due to the very low precision and recall achieved on a number of particularly small files (containing only a handful of clean words). While micro-averaged values obviously give a better indication of the practical usefulness of the system, the macro-averaged scores can be used to establish whether observed differences are statistically significant.

	micro-averaged			macro-averaged		
	F-score	precision	recall	F-score	precision	recall
Baseline	91.23	84.16	99.58	85.32 ± 15.11	79.00 ± 20.14	96.74 ± 02.13
Heuristics only	89.39	85.12	94.11	84.06 ± 18.03	79.16 ± 20.61	94.11 ± 15.02
NCLEANER	91.60	91.95	91.25	87.63 ± 15.50	88.47 ± 14.21	89.74 ± 16.83
Non-lexical model	89.99	90.17	89.82	85.60 ± 16.84	85.35 ± 17.53	89.37 ± 17.37

Table 2: Internal evaluation results of the NCLEANER algorithm on the gold standard of Bauer et al. (2007), showing both micro-averaged and macro-averaged F-score, precision and recall (as percentages calculated at word level).

It is obvious from Table 2 that NCLEANER achieves an improvement over the baseline, and that the n-gram language models are an essential component of the algorithm: using only the heuristic rules, performance drops even below the baseline. The macro-averaged data show that the improvement is statistically significant (paired t-test for F-scores of NCLEANER vs. baseline across texts: $t = 1.976$, df $= 157$, $p = .0499$).

More importantly, NCLEANER trades off recall for precision, achieving a good balance with both precision and recall above 91%. Note that the increase in micro-averaged precision from 84.16% to 91.95% corresponds to a 50% reduction of the error rate. Since it is easy to collect large amounts of text from the Web, high recall is usually not important for Web corpora, so the trade-off is well justified.

The histogram in Figure 3 shows that precision is almost always increased over the baseline for the individual files in the gold standard, so users of NCLEANER can be confident that automatically cleaned Web pages will only rarely be "worse" than their uncleaned counterparts. For a considerable number of files, precision was increased by more than 10 percent points.

Of course, the non-lexical model does not perform as well, but still achieves significantly better precision than the baseline (paired t-test for precision across texts: $t = 9.0557$, df $= 157$, $p = 5.055 \cdot 10^{-16}$).

5. External evaluation

The standard parameter files included in the NCLEANER distribution were trained on the full gold standard of Bauer et al. (2007). Further evaluation of these models was carried out on an external data set: the official test set of the CLEANEVAL competition, containing 674 manually cleaned files with a total of 1.6 million words of clean text.

The micro-averaged results reported in Table 3 give a good indication of the accuracy that NCLEANER will achieve in real-world applications. It is also interesting to compare the intuitively meaningful precision and recall values shown there to the official overall score that NCLEANER achieved on the same test set in the CLEANEVAL competition (Table 1).

Quite surprisingly, the trade-off between recall and precision works even better on the external data set, where the NCLEANER system achieves a micro-averaged precision of 94.70%, corresponding to a reduction in error rate by almost 70%. Note that the baseline results are much lower than for the internal evaluation, with a recall of little more than 95%. This can mostly be traced to some issues with character encoding, formatting and "hidden" text in the Web pages. Apparently, the text dumps that served as a basis for the manual cleanup of the CLEANEVAL test set were generated by a different tool than Lynx.

	F-score	precision	recall
Baseline	88.72	83.11	95.15
NCLEANER (HTML)	92.73	94.70	90.83
NCLEANER (text)	90.18	90.30	90.05
Non-lexical (HTML)	92.31	91.65	92.97
Non-lexical (text)	89.86	89.88	89.85

Table 3: Micro-averaged evaluation results of the standard NCLEANER parameter files on the official CLEANEVAL test set (percentages calculated at word level).

When applied to existing text dumps rather than HTML files, NCLEANER achieves considerably lower accuracy (demonstrating the importance of the heuristics, which seemed only to make text quality worse in the cross-validation experiments). It is still a significant improvement over the baseline with well-balanced precision and recall of approximately 90%.

The non-lexical model performs somewhat worse than the standard model on HTML files, especially with respect to precision. On plain text dumps, there is no significant difference between the two models. At least for this purpose, the non-lexical parameter file contained in the NCLEANER distribution is probably also useful for other European languages than English.

	F-score	precision	recall
Baseline	0.00	0.00	0.00
NCLEANER (HTML)	60.85	69.30	54.24
NCLEANER (text)	34.04	28.98	41.25
Non-lexical (HTML)	55.76	52.76	59.13
Non-lexical (text)	33.21	27.43	42.07

Table 4: Micro-averaged labelled segmentation accuracy of the standard NCLEANER parameter files (official CLEANEVAL test set, percentages of all segment markers).

Table 4 shows evaluation results for paragraph segmentation and labelling (as heading, list item or standard text paragraph). The F-score of the standard NCLEANER model applied to HTML pages is remarkably similar to its official result in the CLEANEVAL competition (Table 1). While

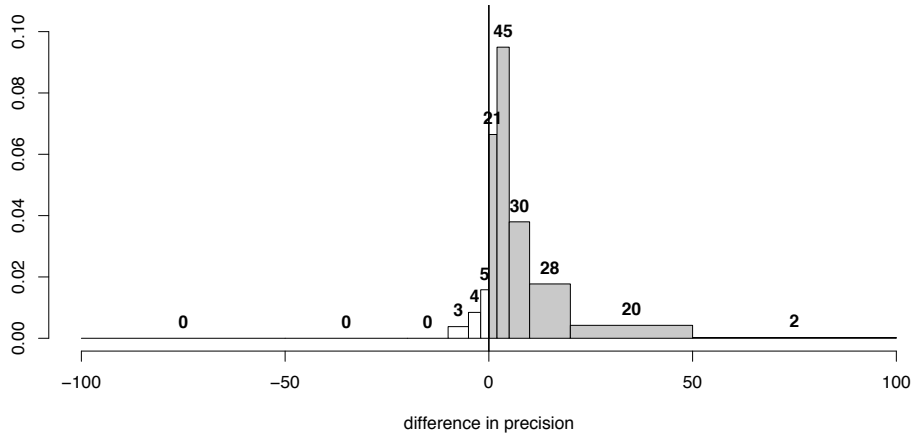

Figure 3: Distribution of precision differences between NCLEANER and the baseline for individual files in the gold standard of Bauer et al. (2007).

NCLEANER does not achieve satisfactory accuracy in this task yet, the main focus of future development will remain on text cleaning. Without information from the HTML markup (i.e. on plain text file), labelling performance is random (almost all segments are identified as standard paragraphs).

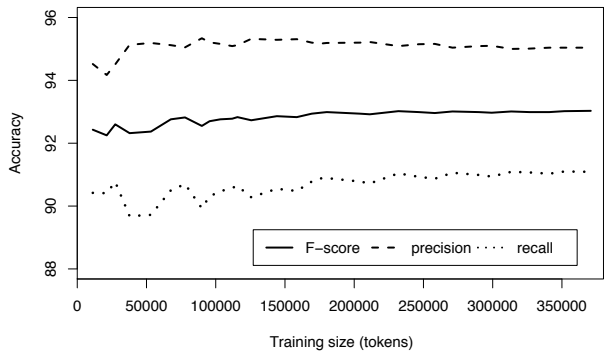

Figure 4: Learning curve of NCLEANER, using incremental subsets of the gold standard (Bauer et al., 2007) for training and the CLEANEVAL test set for evaluation (all values are micro-averaged).

A crucial issue for any supervised learning approach is the amount of training needed to train a high-quality model. Figure 4 shows the *learning curve* of NCLEANER trained on incremental subsets of the gold standard. The F-score curve quickly levels off, and precision even degrades slightly due to overtraining effects. While there is some random variation for very small training sets, 50,000 to 100,000 words of training data appear to be completely sufficient, corresponding to less than 50 manually cleaned Web pages. Even with a very small training set consisting of 10 or fewer randomly selected Web pages, precision is consistently above 94% and recall around 90%. This suggests that NCLEANER can be adapted to other languages or tasks (such as cleaning newsgroup postings) with minimal amounts of manually annotated data.

6. Availability and practical issues

One advantage of the naive n-gram modelling approach taken by NCLEANER is its low computational complexity. The size of the standard parameter file is only 2.3 MB in a redundant, human-readable format; in compressed form, it shrinks to 600 KB. Even the research prototype implemented in fully portable Perl code processes approximately 20 million words per hour on a standard server-class machine and occupies less than 20 MB of working memory while running. The mixed Perl/C implementation processes more than 120 million words per hour, making it suitable even for large Web corpora in the gigaword range.

The NCLEANER software is available as an open-source Perl package (`Text-NCleaner`), which can be downloaded from `http://webascorpus.sf.net/`. The Python script used for evaluation (`cleaneval.py`) is available from the same Web page.

7. References

Michele Banko and Eric Brill. 2001. Scaling to very very large corpora for natural language disambiguation. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics*, pages 26–33, Toulouse, France.

Marco Baroni and Silvia Bernardini, editors. 2006. *Wacky! Working papers on the Web as Corpus*. GEDIT, Bologna. Online version: `http://wackybook.sslmit.unibo.it/`.

Daniel Bauer, Judith Degen, Xiaoye Deng, Priska Herger, Jan Gasthaus, Eugenie Giesbrecht, Lina Jansen, Christin Kalina, Thorben Krüger, Robert Märtin, Martin Schmidt, Simon Scholler, Johannes Steger, Egon Stemle, and Stefan Evert. 2007. FIASCO: Filtering the Internet by automatic subtree classification, Osnabrück. In *this volume*.

Adam Kilgarriff and Gregory Grefenstette. 2003. Introduction to the special issue on the Web as corpus. *Computational Linguistics*, 29(3):333–347.

Christopher D. Manning and Hinrich Schütze. 1999. *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge, MA.