Building Large Corpora from the Web Using a New Efficient Tool Chain

Roland Schäfer, Felix Bildhauer

German Grammar, SFB 632/A6
Freie Universität Berlin
Habelschwerdter Allee 45, 14195 Berlin
roland.schaefer@fu-berlin.de, felix.bildhauer@fu-berlin.de

Abstract

Over the last decade, methods of web corpus construction and the evaluation of web corpora have been actively researched. Prominently, the WaCky initiative has provided both theoretical results and a set of web corpora for selected European languages. We present a software toolkit for web corpus construction and a set of siginificantly larger corpora (up to over 9 billion tokens) built using this software. First, we discuss how the data should be collected to ensure that it is not biased towards certain hosts. Then, we describe our software toolkit which performs basic cleanups as well as boilerplate removal, simple connected text detection as well as shingling to remove duplicates from the corpora. We finally report evaluation results of the corpora built so far, for example w. r. t. the amount of duplication contained and the text type/genre distribution. Where applicable, we compare our corpora to the WaCky corpora, since it is inappropriate, in our view, to compare web corpora to traditional or balanced corpora. While we use some methods applied by the WaCky initiative, we can show that we have introduced incremental improvements.

Keywords: web corpora, corpus evaluation, web crawling

Corpus	N Documents	N Tokens
SECOW2011	1.91	1,750
ESCOW2012	1.30	1,587
DECOW2012	7.63	9,108

Table 1: Document and token counts (million) for selected COW corpora

1. Introduction

In this paper, we describe a new tool chain for the processing of web data for the purpose of web corpus construction. Furthermore, a growing set of very large web corpora, which we constructed using the aforementioned tools, is introduced and evaluated (Table 1 lists a selection of these corpora and their respective sizes). We proceed by discussing the methods of data collection in 2., describing the methods implemented in the software in 3. before reporting results of the evaluation of the final corpora and the quality of the software in 4.

2. Data Collection

The purpose of the project described here is the construction of very huge comparable web corpora in a large number of languages. Although we share with Kilgarriff and Grefenstette (2003, p. 340) a more or less agnostic position towards the question of representativeness of web corpora, we still consider it crucial to define the population of which a corpus is intended to be a sample. We sample from all WWW documents under a given national TLD, written in an official language of the respective country and containing predominantly connected text. From each document, we attempt to extract all paragraphs of connected text, finally making sure that the majority of the text within the document occurs only once in the final corpus (cf. 3.4.). Since the given population is unknown to a large extent, proportional sampling is out of the question, and large random samples are the only valid option. How a random sample from the web can be approximated is not a settled question. In web corpus construction, it is customary to send large numbers of requests to search engines using tuples of mid-frequency word forms as search strings (Baroni and Bernardini, 2004). Either only the URLs returned are then harvested (BootCaT method), or they are used as seed URLs for a crawler system. We apply the second method, since corpora of roughly 10^{10} tokens cannot practically be constructed from search engine queries alone. Also, the BootCaT method, as opposed to an approach using specialized crawler software, does not allow to effectively control the politeness of the harvesting process (Manning et al., 2009, 444ff.). As a crawler we use Heritrix 1.4 configured roughly as described by Emerson and O'Neil (2006). Seed URLs were taken from Yahoo until their discontinuation of the free API access, after which we switched to Microsoft Bing. Besides the fact that the Yahoo API discontinuation proves that total dependence on search

engines is generally an unsafe strategy, we would like to point out some problems with search engine results and argue for much stronger reliance on very large web crawls for linguistic purposes.

The randomness of web corpora was examined successfully in the BootCaT/WaCky community from a linguistic viewpoint. For example, Ciamarita and Baroni (2006) evaluate randomness of corpora based on linguistic features. The most basic measure of randomness of both seed URLs retrieved from a search engine and documents downloaded by a crawler, however, is its bias towards certain hosts. Even though a web corpus (or a set of search engine results) might never be called balanced, it should count as biased if it proportionally contains a huge number of documents from only a few hosts. It is known from papers such as Bar-Yossef and Gurevich (2006) that obtaining random samples from search engines is not a simple matter, especially when there is only one major search engine left which allows free API access for large numbers of queries. To our knowledge, nobody has ever constructed web corpora making efforts such as described by Bar-Yossef and Gurevich (2006). Since the method described in that paper only approximates random samples from search engine indices and not from the web itself, even following the procedure would not ensure random samples in the desired sense. The customary method, however, does not even ensure bias-free sampling from a search engine's index.

Our experiments so far have shown that a large number of hosts which never occur in search engine results can be discovered through long-term crawling. In Figures 1 and 2, experiments for the `de` and `se` domain are evaluated with respect to their host bias. The graphs cumulatively plot for a random sample of 10,000 URLs from either a set of unique seeds gathered from the search engines (Yahoo for `se`, Bing for `de`) and the document URLs of a final corpus derived from these seeds (which has undergone the cleaning procedures explained in 3.) how large a proportion r of the total URLs comes from to the n most popular hosts in the whole sample. The seed URLs were queried using 3-tuples of content word forms (nouns, verbs, adjectives, adverbs) ranked 1,001 through 6,000 in frequency lists either from existing corpora or from previous crawl experiments. A maximum of 10 results per tuple were requested from the search engine. The crawler ran for approximately seven days for the `se` domain, and for 28 days in the `de` case. The total sizes of the seed sets N_{seeds} and the total number of document URLs in the corpus N_{corpus} is given for each experiment. The third curve in each graph plots the proportion of documents in the final corpus which stem from the top n hosts from the seed set. It is obvious that the experiments were successful to different degrees.

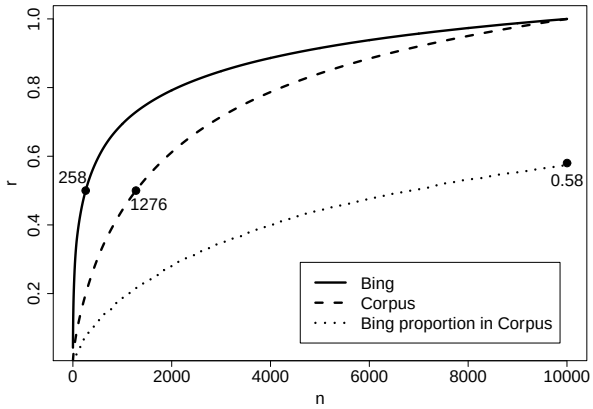

Figure 1: Host bias for DECOW2012 with Bing seeds, $N_{corpus} = 7,759,892$, $N_{seeds} = 912,243$

Using an extremely large set of unique seed URLs (over 900,000) causes the seed set to be comparatively less biased: in the DECOW2012 seeds (Figure 1), 50% of the URLs come from the 258 most popular hosts, while in the SECOW2011 (Figure 2) seed set, figures is much lower: 29 hosts contribute 50% of the seed URLs.[1] A longer crawling time helped to bring down the host bias of DECOW2012 significantly, such that in the final corpus, 50% of the documents in the sample come from 1,276 hosts. Also, the 10,000 most prominent hosts in the DECOW2012 seed set contribute only 58% of the documents from the top 10,000 hosts in the final corpus.

The graph for SECOW2011 (Figure 2) in fact shows a completely unsuccessful crawl experiment, where an already heavily biased seed set resulted in even more biased crawl results. The seeds as returned by Yahoo were extremely biased to begin with, and the crawl, using a breadth-first strategy, got stuck trying to exhaustively harvest the few hosts strongly represented in the seeds. This resulted in an interesting corpus, but definitely not in a general-purpose corpus. It contains 1,443,452 documents (75.5% of the total documents) from the host `blogg.se`. Since in this case, the host is strongly tied to a certain type/genre of text and communication (blog posts), it is obvious that host bias can be directly linked to linguistic biases.

To illustrate further that the seed set and the crawling process both play a role, we now compare the re-

[1] Due to the shutdown of Yahoo's API access, it is now impossible to find out whether this is also due to different biases between Yahoo and Bing.

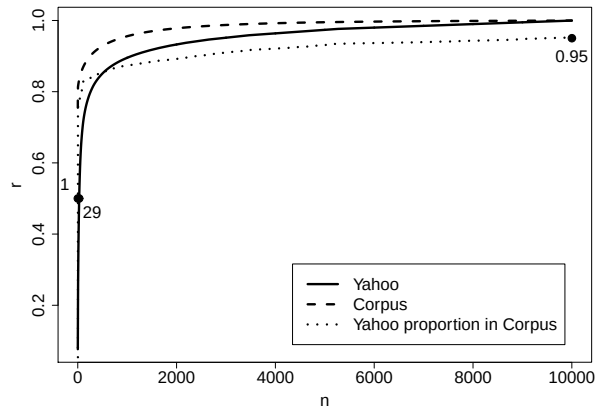

Figure 2: Host bias for SECOW2011 with Yahoo seeds, $N_{corpus} = 1,912,757$, $N_{seeds} = 204,872$

sults to a version of the deWaC corpus from which we removed near-duplicates using our shingling tools (as described in 4.2.) in Figure 3. Only 8,631 hosts are considered, because the seed set contains only this many hosts. The deWaC crawl was based on a seed set where each host was represented only once, resulting in a linear host bias graph. During the 10 day crawl using a breadth-first strategy, however, the crawler harvested mainly the hosts from the seed set, leading to a much stronger bias. This corroborates our view that both the seed sets and the crawling strategies have to be chosen with greater care than is customary.

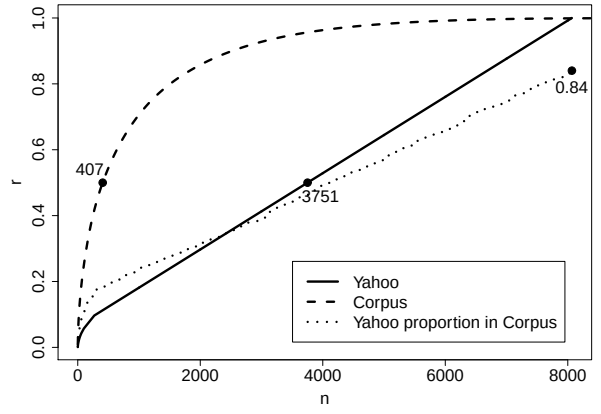

Figure 3: Host bias for shingled deWaC with Yahoo seeds, $N_{corpus} = 1,501,076$, $N_{seeds} = 8,631$

Finally, host bias is reflected also in the average number of documents per host, which is 20 for DECOW2012, 30 for ESCOW2012 and 158 for the shingled deWaC. Since we can safely assume that the population of hosts in the respective TLDs is much larger than the number of hosts represented the corpora, we consider strong host biases unacceptable.

In our view, these results allow at least the following preliminary conclusions:

1. Relying solely on search engine queries might introduce severe bias towards certain hosts and thus to certain types of document or register. Relying on search engines is also unwise because it is foreseeable that no search engine provider will allow free API access in the long run.
2. Long, extensive web crawls can help to alleviate a host bias in the seeds.
3. Crawler software should be adapted or rewritten to offer crawling strategies optimized for better randomness of corpus crawls. Simply implementing a per-host quota in the crawler (a limit to n documents per host) is not a wise idea, since the limit should be imposed on n good documents coming from a host, but the quality of the documents can currently only be assessed after post-processing.

Since specific crawler software for corpus crawls has not been written, and our own crawler is still in planning stage, we suggest to provisionally use existing crawler software and doing very long crawls to create corpora like DECOW2012.

3. Post-Processing

In this section, we describe the function of our own tool chain, which fills the gap between the crawler and linguistic post-processing and indexing tools. The input is typically a file (or directory containing multiple files) which contain(s) HTML/XML documents. The tools are preconfigured for ARC files as created by the Heritrix 1.4 crawler, but they can be reconfigured to deal with alternative file formats and to apply external pre- and post-processing commands to the input files. They are written in a modern variant of ObjectPascal[2], which also allows for parallelization of the more demanding algorithms.

3.1. Basic Cleanup

Under basic cleanup, we subsume the trivial tasks of HTML and script removal, codepage conversion, etc. as performed by the *texrex* tool. HTML input is read from a stream, parsed on the fly, cleansed from markup and scripts, and buffered for further processing. Since the tool chain is optimized for ISO-8859

[2] `http://www.freepascal.org/`

languages, a simple conversion from UTF-8 multibyte characters to corresponding ISO-8859 characters as well as a complete HTML entity conversion is also performed on the fly. The software deals with faulty markup by favoring cleanliness of output over conservation of text and discards the whole document in case of serious unrecoverable errors.

Based on a configurable list of markup tags, a simple paragraph detection is performed, inserting paragraph breaks wherever one of the configured tags occurs. Optionally, the results of basic cleansing are then subjected to a second pass of markup removal, since many web pages contain literal markup pieced together with entities (`<br>` as `&lt;br&gt;` etc.). Perfectly identical subsequent lines are also removed, and excessively repeated punctuation is compressed to reasonable sequences of punctuation tokens.

3.2. Boilerplate Removal

The removal of navigational elements, date strings, copyright notices, etc. is usually referred to as boilerplate removal. The WaCky method (Baroni et al., 2009), simply speaking, selects the window of text from the whole document for which the ratio of text-encoding vs. markup-encoding characters is maximal. This has the advantage of selecting a coherent block of text from the web page, but also has the disadvantage of allowing intervening boilerplate to end up in the corpus. Also, many web pages from blog and forum sites contain several blocks of text with intervening boilerplate, of which many can be lost if this method is applied.

Therefore, we decided to determine for each paragraph individually whether it is boilerplate or not. The decision is made by a multi-layer perceptron as implemented in the FANN library (Nissen, 2005). The input to the network is currently an array of nine values which are calculated per paragraph:

1. the ratio of text encoding characters vs. markup encoding characters,
2. the same ratio for a window extended by one paragraph in both directions,
3. the same ratio for a window extended by two paragraphs in both directions,
4. the raw number of text (non-markup) characters,
5. the ratio of uppercase vs. lowercase characters,
6. the ratio of non-letters vs. letters in the text,
7. the same ratio for a window extended by one paragraph in both directions,
8. the same ratio for a window extended by two paragraphs in both directions,
9. the percentile of the paragraph within the text mass of the whole document.

It is obvious that value 5 is language-dependent to some extent. However, after training, the feature turned out to be weighted so lightly that the network performed equally well for English, German, French, Spanish, and Swedish and we decided to use one network for all languages. This pre-packaged network was trained on decisions for 1,000 German paragraphs coded by humans using the graphical tool *textrain* included in the package. As the only coding guideline which turned out to be practical, we defined:

1. Any paragraph containing coherent full sentences is text,
2. any heading for such text (i. e., not page headers etc.) is text,
3. everything else is boilerplate.

Networks can be trained with user-coded data using the included *texnet* tool. The network must use a symmetric Gaussian output activation function, which produces real numbers between 0 and 1. The user can then evaluate the results and determine a cutoff point in $[0..1]$ below which a paragraph will actually be considered boilerplate in a production run.[3] The measures of accuracy for the pre-packaged network depending on the selected cutoff are shown in Figure 4, which measures the results of an application of the pre-packaged network to 1,000 unseen German paragraphs coded using identical guidelines. The overall accuracy is quite adequate compared to the approaches described in (Baroni et al., 2008). For the COW corpora, we selected the cutoff with the highest F-score (harmonic mean of precision and recall), which is 0.48.

3.3. Detecting Connected Text

Language identification on the document level has been around since the 1960s, and detecting a known language is a simple text-book matter. Since our corpora are designed mainly for research in theoretical linguistics, we were not satisfied with simple language detection, for example based on character n-grams. Many documents on the web contain some text in a specific language, but it is often not connected text but tables, tag clouds, or any other non-sentence material. For example, we ran lists of German content words without function words (artificial tag clouds) through standard language identifiers such as the one included

[3] The *texrex* tool provides functions to create evaluation output where decisions are logged but not executed.

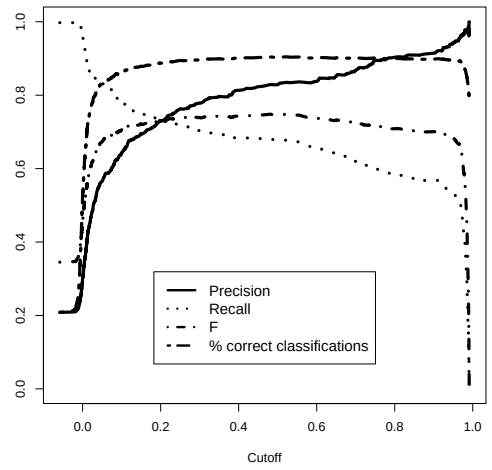

Figure 4: Quality of boilerplate removal depending on cutoff value (pre-packaged network on 1,000 unseen German paragraphs)

in the LingPipe software.[4] For any such list, it recognized German with $P(Category|Input) = 1$. This is clearly undesirable for the intended kind of corpus.

The WaCky corpora were cleaned by allowing only documents which contain ten types and 30 tokens of function words from the target language (Baroni et al., 2009). This is simple and effective in detecting connected text, but it makes the result depend heavily on the length of the input documents and fails to remove long documents with a mix of languages of which the target language represents a large but still not the largest portion. We use the same approach, but without the dependence on absolute token frequencies of certain types. A separate tool (*texprof*) is available which generates a profile for a given language. It takes a specifiable number m of training documents and applies a minimal tokenizer to them (anything in between two non-letters in ISO-8859 is a token). For a specifiable number n of the most token-frequent types $t_{1,..,n}$ across all training documents, it calculates the weighted arithmetic mean of the relative frequencies of these types and the weighted standard deviation (weighted by document length). If $\mu(t)$ is the weighted mean for type t in the training documents, $\sigma^2(t)$ the corresponding weighted standard deviation, and $f(t, d)$ the relative frequency of t in d, then we can (in a production run) for each document d calculate a standardized summed negative deviation $B(d)$ (from the mean in the training documents) for all t_n.

$$z(t, d) = \frac{\mu(t) - f(t, d)}{\sigma^2(t)} \quad (1)$$

$$b(t, d) = \begin{cases} z(t, d) & \text{if } z(t, d) > 0 \\ 0 & \text{else} \end{cases} \quad (2)$$

$$B(d) = \sum_{i=1}^{n} b(t_i, d) \quad (3)$$

Finally, all documents are removed for which $B(d)$ exceeds a certain threshold. As training documents for the COW corpora, we used a hand-selected subset of the downloaded documents with $m \approx 400$, $n = 10$. We discarded documents with $B(d) > 10$.

3.4. Perfect and Near Duplicate Removal

Duplicate removal is done in two steps. The *texrex* tool performs perfect duplicate removal by creating for each document an array of 128 characters, which are evenly distributed over the whole document. These arrays are stored using a fast string hash table in memory, and documents corresponding to known arrays are discarded.

To perform near duplicate removal, there is a separate tool chain providing a native implementation of w-shingling (Broder et al., 1997) without clustering. Documents are tokenized by the parallelized *teshi* tool, which then forms the set of token-n-grams (or w-shingles) for each document. The shingles are hashed with m 64-bit Rabin hash functions (Rabin, 1981), each seeded with a 64-bit representation of a different irreducible polynomial to approximate the random permutations of the hashes required according to Broder et al. (1997). The document's w-shingling is the set of minimal hashes for each m. The *tender* tool then sorts the shingle database and calculates the number of shingles shared between each document. From document pairs with a number of shared shingles higher than a controlable threshold $t \cdot m$, the shorter one is finally removed by a separate tool. For the COW corpora: $n = 5$, $m = 100$, $t = 0.05$.

3.5. Software Performance

The performance of the software is major a aspect in the construction of very large corpora. For DECOW2012, *texrex* (cf. sections 3.1. to 3.3. plus perfect duplicate removal), which is not parallelized so far, ran for 8.8 CPU days on a Xeon 5160 at 3.00 GHz, processing 130,602,410 input documents (170 input documents per second). The process, however, was slowed down due to compression/recompression of the data with *gzip*.

[4] See `http://alias-i.com/lingpipe/`. The method implemented is based on (Tehan, 2000) according to the manual.

The partly parallelized shingling tools ran for a total of 2.3 CPU days (using eight shingling threads) on the same machine as above, processing the 16,935,227 documents left over by the previous tools (85 input documents per second).

4. Corpus Evaluation

4.1. Effect of the Different Algorithms

First, we report the amount of data filtered (as the number of documents discarded) by the different algorithms described in 3. The figures are for DECOW2012, for which a total of 130,602,410 documents was crawled. To keep the processing demands of the more costly algorithms like duplicate detection and boilerplate removal down, we added a simple threshold filter after the basic cleanup (cf. 3.1.), which in the case of the COW2012 corpora caused documents with less than 2,000 characters to be discarded right away.

Algorithm	Documents	Percentage
Threshold	93,604,922	71.67%
Language	16,882,377	12.93%
Perfect dup.	3,179,884	2.43%
Near dup.	9,175,335	7.03%
Total	122,842,518	94.06%

Table 2: Number of documents discarded by the consecutive post-processing algorithms and percentage discarded from the total number of crawled documents (DECOW2012)

4.2. Duplication

Although w-shingling removes more than half of the documents left over from previous post-processing steps (see Table 4), we still have some amount of duplication in our corpora. In order to roughly estimate the magnitude of the remaining duplication, we are considering keywords in context (KWIC), using the following procedure: (1) Choose a frequent word form. (2) Extract all its occurrences from the corpus, plus a context of n characters to its left and right. (3) Count the repetitions. The results reported below were obtained with n = 60, corresponding to approx. 7 – 9 words on either side of the keyword. Thus, the stretch of text is, in almost all cases, larger than the 5-shingles used in duplicate detection. This approach is not directly indicative of the number duplicated documents, both because it is likely to yield more than one repetition in the case where two documents are actually duplicated, and because it will not only count duplicated documents, but also all sorts of quotations occurring in documents, which represent an unavoidable kind of duplication. In short, the method is likely to overestimate the actual amount of duplication rather than to underestimate it. Table 3 shows the amount of duplication that remains in the published versions of both the COW and the WaCky corpora detected by the KWIC method. For each corpus, the three most frequent types were chosen as keywords.

DECOW2012		**DEWAC**	
9.1 G tokens		1.6 G tokens	
keyword	%	keyword	%
und	9.25	*der*	11.40
die	5.70	*die*	11.19
der	6.11	*und*	11.09
ESCOW2012		**FRWAC**	
1,6 G tokens		1.6 G tokens	
keyword	%	keyword	%
de	7.15	*de*	17.38
la	5.99	*la*	17.33
que	7.28	*et*	17.80

Table 3: Duplicated strings: keyword plus 60 characters to left and right

Corpus	Before	After	% Reduction
DECOW2012	16.94	7.63	54.9
ESCOW2012	3.50	1.30	63.0
DEWAC	1.75	1.50	14.3
FRWAC	2.27	1.47	35.0

Table 4: Reduction in millions of documents by w-shingling (5% or higher w-shingling overlap; same criteria applied to the WaCky corpora).

Finally, Table 4 shows the number of documents before and after w-shingling, both for our German (DECOW2012) and Spanish (ESCOW2012) corpora, and for the German and French WaCky corpora. Note that the WaCky corpora had already passed through a weaker process of duplicate detection (Baroni et al., 2009), and the counts shown illustrate the remaining amount of duplication which our tools (cf. 3.4.) removed.

4.3. Quality of Connected Text Detection

The quality of the connected text filter described in 3.3. is difficult to assess. At this point, we can say that as a language identifier, the algorithm can be configured to achieve $Precision = 1$, $Recall = 0.97$,

preliminarily evaluated on 120 randomly selected German Wikipedia documents, 100 German literary texts from the twentieth century, 5 German blog posts, and 15 Wikipedia documents in Dutch, English, and 5 different German dialects. The minimal loss in recall comes from the negative classification of a few of the literary documents which, admittedly, could be considered stylistically marked.

To make the algorithm work as a connected text detector, the threshold has to be lowered considerably, and it then achieves (e. g., for DECOW2012): $Precision > 0.95$, $Recall < 0.8$. These last values were obtained in an evaluation with 1,100 artificial documents consisting of a mix of sentences from connected German text and non-connected content words in varying proportions. These figures, however, are rough estimates, mainly because we lacked a satisfying answer to the question of what defines a "document containing a significant amount of connected text in language L", which is why we chose to use artificial documents. Only with a proper operationalization of this notion would a report of the precision and recall of the connected text filter make true sense. Future work will therefore involve the definition of a gold standard set of training and evaluation documents for each language.

4.4. Text Types and Genres

It is not our goal to build corpora which parallel any existing balanced corpora. Given this goal, providing information on the text type composition of our corpora is rather a matter of characterization than one of evaluation. Although Sharoff (2006) is slightly different in his goals, we build on the taxonomy suggested there, which is itself derived from an EAGLES specification. He examines a set of experimental web corpora and established general-purpose corpora with regard to *authorship*, *mode*, *audience*, *aim*, *domain*. We refer the reader to the paper (Sharoff, 2006, 77ff.) for the details. In a test coding of two random samples of 200 documents from the DECOW2011 test corpus, two coders were satisfied with the available categories for *authorship* (with some necessary clarifications), *mode*, *audience*, and *aim*, but amendments were felt to be necessary for *domain*. We therefore introduced one new value for *mode* and completely re-grouped *domain*. The mode *quasi-spontaneous* is a relabeling of Sharoff's *electronic*. The full revised coding guidelines are omitted here for space reasons, but can be downloaded from the project web page.[5] We are aware that there is an active research community on web genres including interest in automatic detection and involving special fields such as sentiment analysis. However, at this point we are only interested in a preliminary overview of the balance of the major genres in our corpora.

Variable	% Agreement	Cohen's κ
Authorship	89.0	.85
Mode	98.0	.94
Audience	88.0	.64
Aim	73.0	.61
Domain	86.0	.82

Table 5: Inter-rater agreement for document classification categories (DECOW2012, $n = 200$)

A major problem was how to deal with possible multiple categorizations. For example, a blog post can be *aim: information*, but the comments can turn the whole web page into predominantly *aim: discussion*. The problem manifests itself even more strongly in the coding decisions for *domain*. Since such conflicts were unresolvable in the given time frame and for the given purpose, we interpreted the lists of possible values for most categories as hierarchical in the sense that, in case of doubt, the topmost value should be selected. For the very frequent mix of blog posts (*mode: written*) and comments (*mode: quasi-spontaneous*) we introduced the new value *mode: blogmix*. In Table 5, the inter-rater agreement (raw agreement and Cohen's κ) per category for two independent coders of a DECOW2012 sample is shown. Table 6 provides the categorization results for DECOW2012 (percentages are consistently those of one of the coders, namely the first author of this paper) and ESCOW2012 (second author). The sample size was $n = 200$ as suggested by Sharoff (2006), and we explicitly give the confidence intervals at a 90% confidence level. While the inter-rater reliability is very good, the confidence intervals show that taking a larger sample would be preferable.

5. Outlook

Future versions of the software are planned to include, among other things, (i) parallelization of *texrex*, (ii) a minimal crawler system to allow corpus construction without the need for search engines and complex crawler systems like Heritrix, (iii) full Unicode support and codepage conversion (based on the ICU library). We are also planning to report on the quality of existing models of POS taggers and other tools for linguistic post-processing when applied to web corpus data, possibly training new models that are better suited to the kind of data. Finally, better automatic recognition of document structure and richer

[5] `http://hpsg.fu-berlin.de/cow/`

	DECOW2012		ESCOW2012	
Type	%	CI ±%	%	CI ±%
Authorship				
Single, female	**6.0**	2.8	**5.0**	2.5
Single, male	**11.5**	3.7	**16.5**	4.3
Multiple	**36.0**	5.6	**16.5**	4.3
Corporate	**21.0**	4.7	**20.5**	4.7
Unknown	**25.5**	5.0	**41.5**	5.7
Mode				
Written	**71.0**	5.0	**86.0**	4.0
Spoken	**1.0**	3.0	**2.5**	1.8
Quasi-Spont.	**22.5**	4.9	**3.5**	2.1
Blogmix	**4.5**	2.4	**8.0**	3.2
Audience				
General	**75.5**	5.0	**94.0**	2.8
Informed	**17.0**	4.4	**2.5**	1.8
Professional	**7.5**	3.0	**3.5**	2.1
Aim				
Recommend.	**12.5**	3.8	**7.0**	3.0
Instruction	**4.5**	2.4	**6.0**	2.8
Information	**36.0**	5.5	**41.5**	5.7
Discussion	**47.0**	5.8	**44.5**	5.8
Fiction	**0.0**	0.0	**1.0**	1.2
Domain				
Science	**2.5**	1.8	**5.0**	2.5
Technology	**14.0**	4.0	**6.5**	2.9
Medical	**4.5**	2.4	**4.0**	2.3
Pol., Soc., Hist.	**21.5**	4.8	**21.0**	4.7
Business, Law	**10.0**	3.5	**12.5**	3.8
Arts	**8.5**	3.2	**8.5**	3.2
Beliefs	**5.0**	2.5	**3.0**	2.0
Life, Leisure	**34.0**	5.5	**39.5**	5.7

Table 6: Text category/genre distribution in DECOW2012 and ESCOW2012 with 90% confidence interval ($n = 200$)

meta data are also primary goals in our further research.

6. Acknowledgements

We would like to thank Sarah Dietzfelbinger for her help in training neural nets and evaluating the COW corpora. Felix Bildhauer's work on this project is supported by the Deutsche Forschungsgemeinschaft through the Sonderforschungsbereich 632, project A6.

7. References

Ziv Bar-Yossef and Maxim Gurevich. 2006. Random sampling from a search engine's index. In *Proceedings of WWW 2006*, Edinburgh.

Marco Baroni and Silvia Bernardini. 2004. Bootcat: Bootstrapping corpora and terms from the web. In *Proceedings of LREC 2004*, pages 1313–1316.

Marco Baroni, Francis Chantree, Adam Kilgarriff, and Serge Sharoff. 2008. Cleaneval: A competition for cleaning webpages. In *Proceedings of LREC 2008*, pages 638–643, Marrakech. ELRA.

Marco Baroni, Silvia Bernardini, Adriano Ferraresi, and Eros Zanchetta. 2009. The wacky wide web: A collection of very large linguistically processed web-crawled corpora. *Language Resources and Evaluation*, 43(3):209–226.

Andrei Z. Broder, Steven C. Glassman, Mark S. Manasse, and Geoffrey Zweig. 1997. Syntactic clustering of the web. Technical Note 1997-115, SRC, Palo Alto, July 25.

Massimiliano Ciamarita and Marco Baroni. 2006. Measuring web-corpus randomness: A progress report. In Marco Baroni and Silvia Bernardini, editors, *Wacky! Working papers on the Web as Corpus*. GEDIT, Bologna.

Thomas Emerson and John O'Neil. 2006. Experience building a large corpus for chinese lexicon construction. In Marco Baroni and Silvia Bernardini, editors, *Wacky! Working papers on the Web as Corpus*. GEDIT, Bologna.

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

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2009. *An Introduction to Information Retrieval*. CUP, Cambridge.

Steffen Nissen. 2005. Neural Networks made simple. *Software 2.0*, 2:14–19.

Michael O. Rabin. 1981. Fingerprinting by random polynomials. Technical Report TR-CSE-03-01, Center for Research in Computing Technology, Harvard University, Harvard.

Serge Sharoff. 2006. Creating general-purpose corpora using automated search engine queries. In Marco Baroni and Silvia Bernardini, editors, *Wacky! Working papers on the Web as Corpus*. GEDIT, Bologna.

William J. Tehan. 2000. Text classification and segmentation using minimum cross entropy. In *In Proceeding of RIAO*.