Focused Web Corpus Crawling

Roland Schäfer
Freie Universität Berlin
roland.schaefer
@fu-berlin.de

Adrien Barbaresi
ENS Lyon
adrien.barbaresi
@ens.lyon.org

Felix Bildhauer
Freie Universität Berlin
felix.bildhauer
@fu-berlin.de

Abstract

In web corpus construction, crawling is a necessary step, and it is probably the most costly of all, because it requires expensive bandwidth usage, and excess crawling increases storage requirements. Excess crawling results from the fact that the web contains a lot of redundant content (duplicates and near-duplicates), as well as other material not suitable or desirable for inclusion in web corpora or web indexes (for example, pages with little text or virtually no text at all). An optimized crawler for web corpus construction would ideally avoid crawling such content in the first place, saving bandwidth, storage, and post-processing costs. In this paper, we show in three experiments that two simple scores are suitable to improve the ratio between corpus size and crawling effort for web corpus construction. The first score is related to overall text quality of the page containing the link, the other one is related to the likelihood that the local block enclosing a link is boilerplate.

1 Crawl Optimization and Yield Ratios

Optimizing a crawling strategy consists in maximizing its weighted coverage $WC(t)$ at any time t during a crawl (Olston and Najork, 2010, 29), i. e., the summed weight of the documents downloaded until t, where the weight of each crawled document is calculated as a measure of the usefulness of the document relative to the purpose of the crawl. To maximize WC, it is vital to guess the weight of the documents behind harvested links before download, such that documents with potentially lesser weight have a lower probability of being downloaded. So-called focused crawlers (in a broad sense) are designed to maximize WC with respect to some specific definition of document weight, for example when documents with a high search-engine relevance (measured as its PageRank or a similar score), documents about specific subjects, or documents in a specific language are desired (Chakrabarti et al., 1999; Menczer et al., 2004; Baykan et al., 2008; Safran et al., 2012). For our purpose, i. e., web corpus crawling, a document with a high weight can simply be defined as one which is not removed from the corpus by the post-processing tools due to low linguistic quality and/or a document which contributes a high amount of text to the corpus. Recently, an interesting approach to crawl optimization along such lines was suggested which relies on statistics about the corpus yield from known hosts (Suchomel and Pomikálek, 2012). Under this approach, the weight (rather of a whole web host) is taken to be the ratio of good documents from the host remaining in the corpus after a specific post-processing chain has been applied to the documents. Harvested URLs pointing to certain hosts are prioritized accordingly. We follow a similar route like Suchomel and Pomikálek, but look at document-local features instead of host statistics.

Throughout this paper, we refer to the **yield ratio** instead of WC, although they are related notions. We define the yield ratio Y_d for a set D_c of crawled unprocessed documents and a set D_r of retained documents after filtering and processing for inclusion in a corpus, with $D_r \subset D_c$, as:

$$Y_d = \frac{|D_r|}{|D_c|} \quad (1)$$

For example, a document yield ratio $Y_d = 0.21$

means that 21% of the crawled documents survived the cleaning procedure (i. e., were not classified as duplicates or spam, were long enough, written in the target language, etc.) and ended up in the corpus. In order to maximize Y_d, 79% of the documents should not have been downloaded in the first place in this example. A parallel definition is assumed for Y_b for the respective amounts of bytes. The document yield ratio is easier to interpret because the byte yield ratio depends on the amount of markup which has to be stripped, and which might vary independently of the quality of the downloaded web pages.

Obviously, the yield ratio – like the weighted coverage – depends highly on the definition of what a good document is, i. e., what the goal of the crawl is. We assume, similar to Suchomel and Pomikálek's approach, that our tools reliably filter out documents that are interesting documents for inclusion a corpus, and that calculating a yield ratio based on the output of those tools is therefore reasonable.[1]

2 Experiment 1: Seed and Crawl Quality

In this experiment, we examine the correlation between the yield ratio of crawler seed URLs and the yield ratio of short Breadth-First Search (BFS) crawls based on those URLs. We used the Heritrix (1.14) web crawler (Mohr et al., 2004) and an older version of the `texrex` web page cleaning toolkit (Schäfer and Bildhauer, 2012). The tools perform, among other things, boilerplate detection and text quality evaluation in the form of the so-called Badness score (Schäfer et al., 2013). A document receives a low Badness score if the most frequent function words of the target language have a high enough frequency in the document. The Badness score is based on previous ideas from language identification and web document filtering (Grefenstette, 1995; Baroni et al., 2009).

Originally, this experiment was carried out in the context of an evaluation of sources of different seed URLs for crawls. In a preliminary step, we began by collecting seed URLs from various sources:

1. the *DMOZ* directory
2. the *Etools* meta search engine
3. the *FriendFeed* social service aggregator
4. the *identi.ca* social bookmarking service
5. *Wikipedia* dumps

We scraped the content behind the URLs and ran a state-of-the-art language identifier (Lui and Baldwin, 2012) on it in order to obtain language-classified seed URLs (Barbaresi, 2013).[2] We then looked specifically at the following languages associated as the single dominant language with at least one top-level domain (TLD):

1. Dutch (`.nl`)
2. French (`.fr`)
3. Indonesian (`.id`)
4. Swedish (`.se`)

We randomly sampled $1,000$ seed URLs for each of the 20 permutations of seed sources and languages/TLDs, downloaded them and used `texrex` to determine the document yield ratio for the documents behind the $1,000$ seeds. The software was configured to perform boilerplate removal, removal of documents based on high Badness scores, perfect duplicate removal, and deletion of documents shorter than $1,000$ characters (after boilerplate removal). Then, we crawled the respective TLDs, starting the crawls with the $1,000$ seed URLs, respectively. In each crawl, we downloaded 2 GB of raw data, cleaned them, and calculated the document yield ratio using the same configuration of `texrex` as we used for cleaning the seed documents. Figure 1 plots the data and an appropriate linear model.

We see that there is a strong correlation (adjusted $R^2 = 0.7831$) between the yield ratio of the documents behind the seed URLs and the yield ratio of the documents found by using the seeds for BFS crawling. It follows that giving high priority to links from pages which are themselves considered high-quality documents by the post-processing tools will likely lead to more efficient crawling. Since there is no fundamental distinction between initial URL seeds and URLs harvested at a later time during the crawl, this effect is likely to extend to the whole run time of a crawl.

[1] This claim should be backed up by forms of extrinsic/task-based evaluation (Schäfer and Bildhauer, 2013, p. 104 ff). Such an evaluation (in the form of a collocation extraction task) was recently presented for our corpora in work by Stefan Evert (Biemann et al., 2013).

[2] See also Barbaresi, this volume.

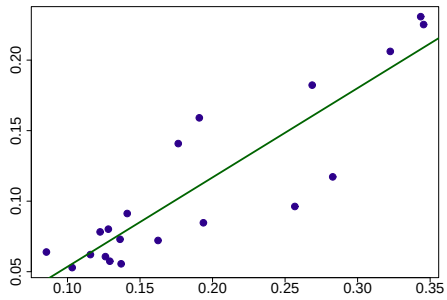

Figure 1: Yield ratio Y_d of the crawls (y axis) plotted against the yield ratio of the documents behind the crawls' 1,000 seeds (x axis). (Higher Y_d is better.) Linear model: $Intercept = -0.0098$, $Coefficient = 0.6332$, $R^2 = 0.7831$ (adjusted), $p < 0.001$ (ANOVA).

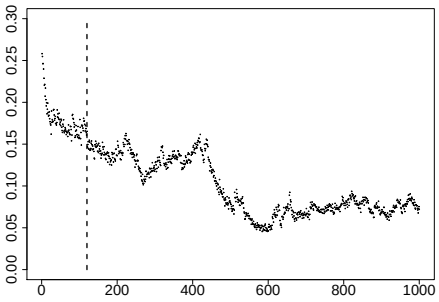

Figure 2: Yield ratio (y axis) over time for a BFS crawl in `.de` in November/December 2011 started with $231,484$ seed URLs scraped from Bing. The yield ratio was calculated at $1,000$ snapshots of 400 MB of data (= one Heritrix ARC file). For snapshots $s_1..s_{500}$: $Y_d = 0.141$, for snapshots $s_{501}..s_{1000}$: $Y_d = 0.071$. The vertical bar marks the point at which the seeds were exhausted. (Schäfer and Bildhauer, 2013, p. 31)

3 Experiment 2: Crawling with Cyclic URL Selection

Using the same configuration of tools as in Section 2, we performed a crawl targeting Flemish documents in the Belgian `.be` national TLD, which hosts both Flemish and French documents in substantial proportions. Usually, even under more favorable conditions (i. e., when we crawl a TLD which contains mostly documents in the target language), the yield ratio of a BFS crawl decreases rapidly in the initial phase, then staying at a low level (Schäfer and Bildhauer, 2013, p. 31). Figure 2 illustrates this with an analysis of a `.de` BFS crawl from late 2011, also processed with the same tools as mentioned in Section 2. Notice that the `.de` domain hosts German documents almost exclusively.

The interesting complication in this experiment is thus the non-target language present in the TLD scope of the crawler and the related question whether, simply speaking, predominantly Flemish documents link to other predominantly Flemish documents rather than French documents. Since the Badness score (calculated as described in Section 2) includes a form of language identification, the yield ratio takes into account this additional complication.

We tested whether the decline of the yield ratio could be compensated for by selecting "high quality" URLs in the following manner: The crawl progressed in five phases. In the first short burn-in phase, we crawled $1,000,000$ documents, and in each of the second to fifth phase, we crawled $10,000,000$ documents. After each phase, the crawl was halted, the crawler frontier was emptied, and the crawl was then re-started with a selection of the URLs harvested in the previous phase. Only those URLs were used which came from documents with a Badness score of 10 or lower (= documents in which the distribution of the most frequent function words fits the expected distribution for Flemish very well, cf. Section 2), and from text blocks with a boilerplate score (Schäfer and Bildhauer, 2012) in $[0.5, 1]$ (= likely not boilerplate). Additionally, it was made sure that no URLs were re-used between the five phases. The very promising results are plotted in Figure 3.

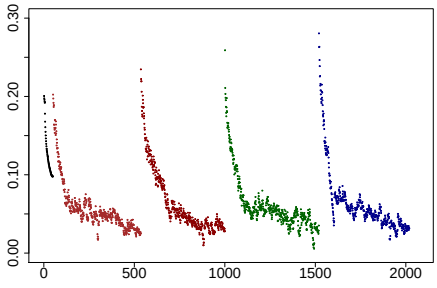

Figure 3: Yield ratio over crawl time with cyclic URL selection in the `.be` TLD. The x axis shows the crawl progression in snapshots of 400 MB of raw crawled data (= one Heritrix ARC file). The y axis shows the yield ratio for each snapshot. The five phases are clearly distinguishable by the sudden increases in yield ratio.

phase	adjusted R^2	p (ANOVA)
1	0.8288	< 0.001
2	0.9187	< 0.001
3	0.8308	< 0.001
4	0.9125	< 0.001
5	0.9025	< 0.001

Table 1: Fit of linear models for the decrease in the yield ratios of the first 100 snapshots in each of the five phases of the `.be` crawl. For the first phase, only 50 snapshots were crawled and fitted.

The decline of the yield ratio is almost linear for the first 100 snapshots in the five phases (cf. Table 1), where each phase has roughly 500 snapshots in total, and one snapshot corresponds to 400 MB of downloaded raw data. After this decline, the yield ratio remains at low levels around 0.05. Cyclic URL selection, however, repeatedly manages to push the yield ratio to above 0.2 for a short period. The subsequent sharp decline shows that link selection/prioritization should rather be implemented in the crawler frontier management in order to achieve a constant effect over longer crawls (cf. Section 5).

4 Experiment 3: Internal Crawl Analysis

For the last experiment, we used the most recent version of the `texrex` toolkit, which writes full link structures for the processed documents as a by-product.[3] An internal analysis of a small portion of a crawled data set from the German TLD was performed, which is part of the raw material of the DECOW corpus (Schäfer and Bildhauer, 2012). The data set contains $11,557,695$ crawled HTML documents and $81,255,876$ `http` links extracted from the crawled documents (only `<a>` tags). Among the link URLs in the sample, $711,092$ are actually links to documents in the sample, so we could analyze exactly those $711,092$ links. It should be noticed that we only looked at links to different hosts, such that host-internal links (navigation to "Home", etc.) are not included in the analysis.

In this experiment, we were interested specifically in the many documents which we usually discard right away simply because they are either very short (below 2 KB of unstripped HTML) or perfect duplicates of other documents. This is a step of document selection which usually precedes the cleansing used for the experiments described in Sections 2 and 3. The analysis shows that of the $711,092$ link URLs in the sample, $130,703$ point to documents which are not perfect duplicates of other documents and which are over 2 KB long. $580,389$ of them point to documents which do not satisfy these criteria. We then evaluated the quality of the link environments in terms of their Badness and boilerplate scores. The results are shown in Figures 4 and 5.[4]

[3] The new version (release name `hyperhyper`) has been released and documented at `http://texrex.sf.net/`.

	positives	negatives
true	$69,273$	$342,430$
false	$237,959$	$61,430$

Table 2: Confusion matrix for binary download decisions based on the Badness of the document containing the URL for the DECOW crawl sample described in Section 4. Badness threshold at 10. Precision=0.225, Recall=0.530, F_1=0.316.

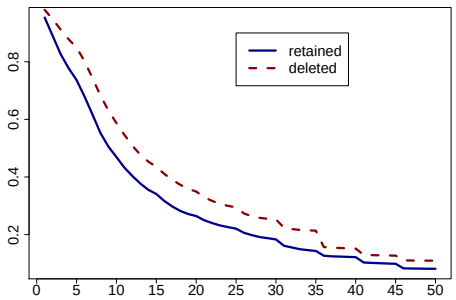

Figure 4: Badness scores of the links in the crawl analysis described in Section 4. The x axis shows the Badness scores of the documents which linked to the retained ("good") and the deleted ("bad") documents. The y axis shows the proportion of retained/deleted documents for which the Badness score is $\geq x$. (Lower Badness scores are better.)

The observable correlation between the quality of a link's context and the quality of the page behind the link is stronger for the boilerplate score than for the Badness score. For example, had we only followed links from documents with a Badness score of 10 or lower (= better), then

[4] Notice that the older version of `texrex` used in the experiments described in Sections 2 and 3 assigns a boilerplate score of 1 to text blocks which are most likely good text, while the new `texrex-hyperhyper` assigns 1 to text blocks which are most likely boilerplate. Take this into account when comparing the thresholds mentioned there and those reported here.

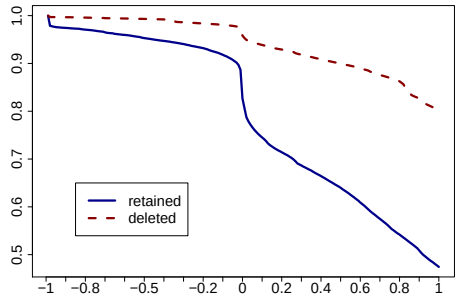

Figure 5: Boilerplate scores of the links in the crawl analysis described in Section 4. The x axis shows the boilerplate scores of the blocks which linked to the retained ("good") and the deleted ("bad") documents. The y axis shows the proportion of retained/deleted documents for which the boilerplate score is $\geq x$. (Lower boilerplate scores are better.)

	positives	negatives
true	83,650	522,350
false	58,039	47,053

Table 3: Confusion matrix for binary download decisions based on the boilerplate score of the block containing the URL for the DECOW crawl sample described in Section 4. Boilerplate threshold at 0.5. Precision=0.590, Recall=0.640, F_1=0.614.

$0.59 \times 580,389 = 342,430$ bad documents would not have been downloaded, but at the same time $0.47 \times 130,703 = 61,430$ good documents would have been lost. Tables 2 and 3 show a confusion matrix for a reasonable Badness threshold (10) and a reasonable boilerplate threshold (0.5). Obviously, if we use Badness and boilerplate scores of the link context to make a binary download decision, the accuracy is much too low, which is why we suggest to merely prioritize URLs instead of discarding them, cf. Section 5.

5 Conclusion and Planned Crawler Architecture

We have shown that two standard cleaning algorithms used in web corpus construction, i. e., text quality evaluation based on frequent short words and boilerplate detection (as implemented in the `texrex` toolkit) have a high potential for optimizing web corpus crawling through the prioritization of harvested URLs in a crawler system.

We are now in the process of designing a custom web corpus crawler system called `HeidiX`, which integrates the `texrex` post-processing tools for weight estimation based on the methods described in this paper. Cf. Figure 6, which schematically shows the current design draft.[5]

`HeidiX` is designed with a system of ranked URL back queues for harvested links (cf. *UrlQueues*). Each queue holds URLs for which the weight estimation is within a specifiable interval, such that the most promising URLs are in one queue, etc. The actual downloading is performed by massively parallel fetcher threads in the *FetcherPool*, which (in the final software) will talk to a DNS cacher and a politeness manager, which handles caching of Robots Exclusion Information and politeness intervals. The fetcher threads pop URLs from one of the ranked queues, which is selected randomly with prior probabilities inversely proportional to the rank of the queue. Thus, promising URLs are popped more often and less promising ones less often.

For guessing the weight, pluggable modules can be used and combined in the *FocusedWalker* container. Currently, we have the standard *UrlSeenFilter*, which is based on our own self-scaling Bloom Filter implementation (Bloom, 1970; Almeida et al., 2007), and which prevents any URL from being queued more than once. We have plans for a URL-based language guesser (Baykan et al., 2008) in the form of the *LanguagePredictor*, and a prioritizer based on the yield from specific hosts as described in Suchomel and Pomikálek (2012) in the form of the *HostYieldPrioritizer*, which reads statistics directly from the `texrex` module. The `texrex` module extracts all hyperlinks from processed documents and tags them with the quality scores described in this paper, such that the *QualityPrioritizer* module can adjust the expected weight of the document behind each URL.

The `HeidiX` architecture also features an alternative queueing strategy in the form of the *RandomWalker*, which allows users to obtain uniform random samples from the web based on existing algorithms (Henzinger et al., 2000; Rusmevichientong et al., 2001). Since obtaining such samples is a goal which is mostly orthogonal to the

[5] Like `texrex`, it is written entirely in the FreePascal dialect of ObjectPascal (`http://freepascal.org/`), uses only very few additional C libraries, and will be released under the GPL 3.

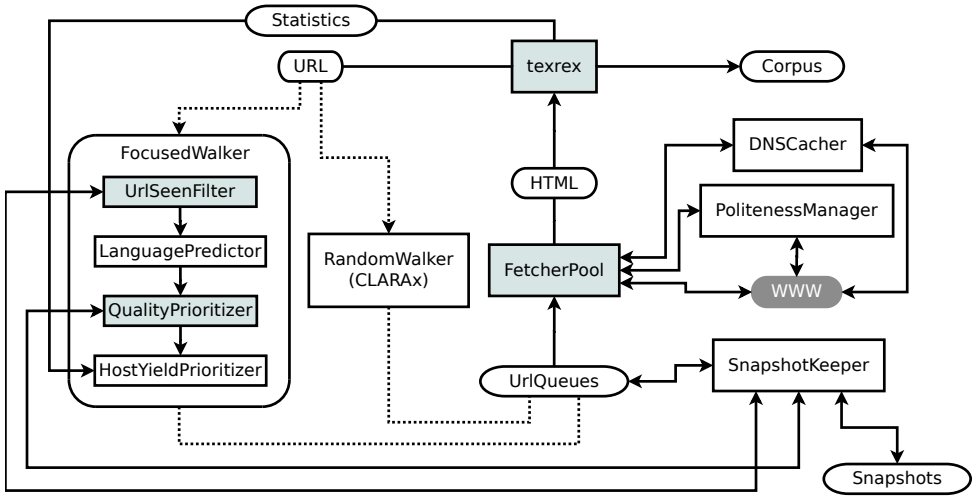

Figure 6: `HeidiX` Crawler Architecture. Grayed modules are done as of March 2014. The FocusedWalker implements an "efficiently locate good corpus document" URL prioritization scheme; the RandomWalker implements bias-corrected Random Walk URL selection for obtaining uniform random samples.

one assumed in this paper, we do not discuss this further here. Finally, a *SnapshotKeeper* module allows users to halt and continue crawls by writing/reading the current state of the relevant components to/from disk.

We hope that `HeidiX` will become a valuable tool in both the efficient construction of very large web corpora (*FocusedWalker*) and the construction of smaller unbiased reference samples as well as web analysis (*RandomWalker*).

References

Paulo Sérgio Almeida, Carlos Baquero, Nuno Preguiça, and David Hutchison. 2007. Scalable bloom filters. *Information Processing Letters*, 101:255–261.

Adrien Barbaresi. 2013. Crawling microblogging services to gather language-classified urls. workflow and case study. In *51st Annual Meeting of the Association for Computational Linguistics Proceedings of the Student Research Workshop*, pages 9–15, Sofia, Bulgaria, August. Association for Computational Linguistics.

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.

Eda Baykan, Monika Henzinger, and Ingmar Weber. 2008. Web page language identification based on URLs. In *Proceedings of the VLDB Endowment*, pages 176–187.

Chris Biemann, Felix Bildhauer, Stefan Evert, Dirk Goldhahn, Uwe Quasthoff, Roland Schäfer, Johannes Simon, Leonard Swiezinski, and Torsten Zesch. 2013. Scalable construction of high-quality web corpora. *Journal for Language Technology and Computational Linguistics*, 28(2):23–60.

Burton Bloom. 1970. Space/time trade-offs in hash coding with allowable errors. *Communications of ACM*, 13(7):422–426.

Soumen Chakrabarti, Martin van den Berg, and Byron Dom. 1999. Focused crawling: a new approach to topic-specific web resource discovery. *Computer Networks*, 31:1623–1640.

Gregory Grefenstette. 1995. Comparing two language identification schemes. In *Proceedings of the 3rd Internation conference on Statistical Analysis of Textual Data (JADT 1995)*, pages 263–268, Rome.

Monika R. Henzinger, Allan Heydon, Michael Mitzenmacher, and Marc Najork. 2000. On near-uniform URL sampling. In *Proceedings of the 9th International World Wide Web conference on Computer Networks: The International Journal of Computer and Telecommunications Networking*, pages 295–308. North-Holland Publishing Co.

Marco Lui and Timothy Baldwin. 2012. langid.py: An Off-the-shelf Language Identification Tool. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (ACL 2012)*, Jeju, Republic of Korea.

Filippo Menczer, Gautam Pant, and Padmini Srinivasan. 2004. Topical web crawlers: Evaluating adaptive algorithms. *ACM Trans. Internet Technol.*, 4(4):378–419.

Gordon Mohr, Michael Stack, Igor Ranitovic, Dan Avery, and Michele Kimpton. 2004. Introduction to Heritrix, an archival quality web crawler. In *Proceedings of the 4th International Web Archiving Workshop (IWAW'04)*.

Christopher Olston and Marc Najork. 2010. *Web Crawling*, volume 4(3) of *Foundations and Trends in Information Retrieval*. now Publishers, Hanover, MA.

Paat Rusmevichientong, David M. Pennock, Steve Lawrence, and C. Lee Giles. 2001. Methods for sampling pages uniformly from the World Wide Web. In *In AAAI Fall Symposium on Using Uncertainty Within Computation*, pages 121–128.

M.S. Safran, A. Althagafi, and Dunren Che. 2012. Improving relevance prediction for focused Web crawlers. In *IEEE/ACIS 11th International Conference on Computer and Information Science (ICIS), 2012*, pages 161–166.

Roland Schäfer and Felix Bildhauer. 2012. Building large corpora from the web using a new efficient tool chain. In Nicoletta Calzolari, Khalid Choukri, Thierry Declerck, Mehmet Uğur Doğan, Bente Maegaard, Joseph Mariani, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, pages 486–493, Istanbul. ELRA.

Roland Schäfer and Felix Bildhauer. 2013. *Web Corpus Construction*. Synthesis Lectures on Human Language Technologies. Morgan and Claypool, San Francisco.

Roland Schäfer, Adrien Barbaresi, and Felix Bildhauer. 2013. The good, the bad, and the hazy: Design decisions in web corpus construction. In Stefan Evert, Egon Stemle, and Paul Rayson, editors, *Proceedings of the 8th Web as Corpus Workshop (WAC-8)*, pages 7–15, Lancaster. SIGWAC.

Vít Suchomel and Jan Pomikálek. 2012. Effcient Web crawling for large text corpora. In Adam Kilgarriff and Serge Sharoff, editors, *Proceedings of the seventh Web as Corpus Workshop*, pages 40–44.