Efficient Web Crawling for Large Text Corpora

Vít Suchomel
Natural Language Processing Centre*
Masaryk University, Brno, Czech Republic
xsuchom2@fi.muni.cz

Jan Pomikálek
Lexical Computing Ltd.
xpomikal@fi.muni.cz

ABSTRACT

Many researchers use texts from the web, an easy source of linguistic data in a great variety of languages. Building both large and good quality text corpora is the challenge we face nowadays. In this paper we describe how to deal with inefficient data downloading and how to focus crawling on text rich web domains. The idea has been successfully implemented in SpiderLing. We present efficiency figures from crawling texts in American Spanish, Czech, Japanese, Russian, Tajik Persian, Turkish and the sizes of the resulting corpora.

Categories and Subject Descriptors

I.2.7 [**Computing Methodologies**]: Artificial Intelligence—*Natural Language Processing*

Keywords

Crawler, web crawling, corpus, web corpus, text corpus

1. INTRODUCTION

Most documents on internet contain data not useful for text corpora, such as lists of links, forms, advertisement, isolated words in tables, and other kind of text not comprised of grammatical sentences. Therefore, by doing general web crawls, we typically download a lot of data which gets filtered out during post-processing. This makes the process of web corpus collection inefficient.

To be able to download large collections of web texts in a good quality and at a low cost for corpora collection managed by SketchEngine[1], we developed SpiderLing—a web spider for linguistics. Unlike traditional crawlers or web indexers, we do not aim to collect all data (e.g. whole web domains). Rather than that we want to retrieve many documents containing full sentences in as little time as possible.

We have experimented with using third party software for obtaining text documents from the web. Following the example of other researchers [2, 3, 1], we have used Heritrix crawler[2] and downloaded documents for the language in interest by restricting the crawl to national web domains of the countries where the language is widely used (e.g. `.cz` for Czech). Though we managed to compile corpora of up to 5.5 billion words in this way [6], we were not satisfied with the fact that we need to keep the crawler running for several weeks and download terabytes of data in order to retrieve a reasonable amount of text. It turned out that most downloaded documents are discarded during post-processing since they contain only material with little or no good quality text.

2. ANALYSIS OF PREVIOUS WORK

We were interested to know how much data we download in vain when using Heritrix and if the sources which should be avoided can be easily identified. In order to get that information we analyzed the data of a billion word corpus of European Portuguese downloaded from the `.pt` domain with Heritrix. For each downloaded web page we computed its yield rate as

$$yield\ rate = \frac{final\ data}{downloaded\ data}$$

where *final data* is the number of bytes in the text which the page contributed to the final corpus and *downloaded data* is simply the size of the page in bytes (i.e. the number of bytes which had to be downloaded). Many web pages have a zero yield rate, mostly because they get rejected by a language classifier or they only contain junk or text duplicate to previously retrieved text.

We grouped the data by web domains and computed a yield rate for each domain as the average yield rate of the contained web pages. We visualized this on a scatterplot which is displayed in Fig. 1. Each domain is represented by a single point in the graph.

It can be seen that the differences among domains are enormous. For example, each of the points in the lower right corner represents a domain from which we downloaded more than 1 GB of data, but it only yielded around 1 kB of text. At the same time, there are domains which yielded more than 100 MB of text (an amount higher by five orders of magnitude) from a similar amount of downloaded data. These

*`http://nlp.fi.muni.cz/`
[1]`http://sketchengine.co.uk/`
[2]`http://crawler.archive.org/`

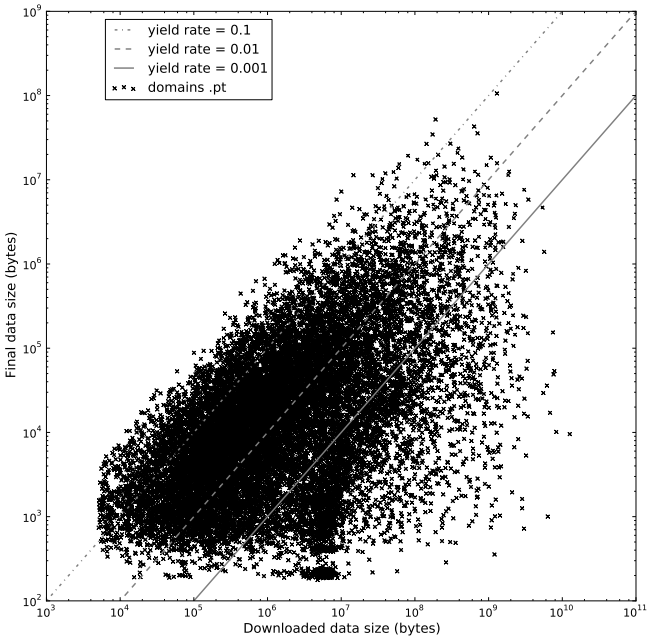

Figure 1: Web domains yield rate for a Heritrix crawl on `.pt`

domains are positioned in the upper right corner of the graph.

Next, we selected a set of yield rate thresholds and computed for each threshold the number of domains with a higher yield rate and the sum of downloaded and final data in these domains. The results can be found in Table 1.

It is easy to see that as the yield rate threshold increases the size of the downloaded data drops quickly whereas there is only a fairly small loss in the final data. This suggests that by avoiding the domains with low yield rate a web crawler could save a lot of bandwidth (and time) without making the final corpus significantly smaller.

For instance if only domains with a yield rate above 0.0128 were crawled, the amount of downloaded data would be reduced from 1289 GB to 86 GB (to less than 7 %) while the size of the final data would only drop from 4.81 GB to 3.62 GB (73.7 %). This is of course only a hypothetical situation, since in practice one would need to download at least several pages from each domain in order to estimate its yield rate. Nevertheless, it is clear that there is a lot of room for making the crawling for web corpora much more efficient.

We observe many web domains offer documents of a similar type. For example, a news site contains short articles, a blog site contains blog entries, a company presentation site contains descriptions of the goods sold or products manufactured. We believe the quality of several documents (with regard to building text corpora) on such sites could represent the quality of all documents within the given domain.

One could argue that a segmentation by domains is too

Table 1: Sums of downloaded and final data size for all domains above the given yield rate threshold

yield rate threshold	domains above the threshold	crawler output size [GB]	final data size [GB]	final yield rate
none	51645	1288.87	4.91	0.0038
0	31024	1181.56	4.91	0.0042
0.0001	29580	705.07	4.90	0.0069
0.0002	28710	619.44	4.89	0.0079
0.0004	27460	513.86	4.86	0.0095
0.0008	25956	407.30	4.80	0.0118
0.0016	24380	307.27	4.68	0.0152
0.0032	22325	214.18	4.47	0.0209
0.0064	19463	142.38	4.13	0.0290
0.0128	15624	85.69	3.62	0.0422
0.0256	11277	45.05	2.91	0.0646
0.0512	7003	18.61	1.98	0.1064
0.1024	3577	5.45	1.06	0.1945
0.2048	1346	1.76	0.54	0.3068
0.4096	313	0.21	0.10	0.4762

coarse-grained since a domain may contain multiple websites with both high and low yield rates. Though we agree, we believe that identifying more fine-grained sets of web pages (like a text rich discussion forum on a text poor goods presentation site) introduces further complications and we leave that for future work.

3. SPIDERLING

Simple web crawlers are not robust enough to suit our needs (e.g. not supporting heavily concurrent communication, lacking load balancing by domain or IP address, not able to restart the crawling after a system crash). On the other hand, the source code of sophisticated crawlers is too complex to alter, making implementation of our way of efficient web traversing difficult.

We came to the conclusion that the easiest way of implementing our very specific requirements on web crawling is to create a custom crawler from scratch. In order to reduce the amount of unwanted downloaded content, the crawler actively looks for text rich resources and avoids websites containing material mostly not suitable for text corpora. Our hope was that by avoiding the unwanted content we can not only save bandwidth but also shorten the time required for data postprocessing and building a web corpus of given size.

3.1 Improving the yield rate

Our primary aim is to identify high-yielding domains and to avoid low-yielding ones. At the same time we want to make sure that we do not download all data only from a few top-yielding domains so that we achieve a reasonable diversity of the obtained texts.

We collect information about the current yield rate of each domain during crawling the web. If the yield rate drops below a certain threshold we blacklist the domain and do not download any further data from it. We define a minimum

Table 2: The yield rate threshold as a function of the number of downloaded documents

documents count	yield rate threshold
10	0.00
100	0.01
1000	0.02
10000	0.03

amount of data which must be retrieved from each domain before it can be blacklisted. Current limit is 8 web pages or 512 kB of data, whichever is higher. The yield rate threshold is dynamic and increases as more pages are downloaded from the domain. This ensures that sooner or later all domains get blacklisted, which prevents overrepresentation of data from a single domain. Nevertheless, low-yielding domains are blacklisted much sooner and thus the average yield rate should increase.

The yield rate threshold for a domain is computed using the following function:

$$t(n) = 0.01 \cdot (\log_{10}(n) - 1)$$

where n is the number of documents downloaded from the domain. The function is based partly on the authors' intuition and partly on the results of initial experiments. Table 2 contains a list of thresholds for various numbers of downloaded documents.

We experimented with various parameters of the yield rate threshold function. Fig. 2 shows how the average yield rate changes in time with different yield rate threshold functions. These experiments have been performed with Czech as the target language. It can be seen that stricter threshold functions result in higher average yield rate. However, too high thresholds have a negative impact on the crawling speed (some domains are blacklisted too early). It is therefore necessary to make a reasonable compromise.

Note: We used the threshold functions from Fig. 2 in our initial experiments. We selected an even less strict one (defined in this section) later on during crawling various data sources. It was a matter of balancing high yield rate versus total amount of obtained data. Too much data was thrown away due to a strict threshold. That is why the currently used threshold function is not present in the figure. The main point is that yield rate is strongly affected by the selected threshold function.

3.2 Removing junk and duplicates

We use jusText[3] [5]—a heuristic based boilerplate removal tool—embedded in SpiderLing to remove content such as navigation links, advertisements, headers and footers from downloaded web pages. Only paragraphs containing full sentences are preserved.

Duplicate documents are removed at two levels: (i) original form (text + HTML), and (ii) clean text as produced by jusText. Two correspondent checksums are computed for each web page and stored in memory. Documents with previously seen checksums are discarded. Both kinds of removal are done on-the-fly during the crawling to immediately propagate the currently crawled documents' yield rate into the corresponding domain yield rate. This enables SpiderLing to dynamically react to obtained data.

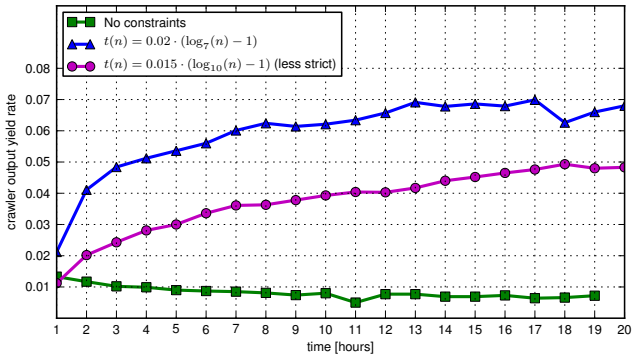

Figure 2: Average yield rate in time for various yield rate threshold functions (crawling the Czech web)

As a post-processing step, we also remove near-duplicates using onion[4]. The deduplication is performed on paragraph level. Paragpraphs consisting of more than 50 % word 7-tuples encountered in previously processed data are removed. Since such deduplication is a higly demanding task in terms of both processor cycles and memory consumption, we did not embed it into SpiderLing. Nonetheless, we are still considering such integration, since it would enable a more accurate estimate of yield rates and thus improve the crawler's traversing algorithm.

We currently do not filter unwanted web content such as link farms and machine generated texts. This may be a subject to further research. Note though that some of such content (e.g. excerpts of Wikipedia articles on link farms) is already reduced in our current processing pipeline as a positive side effect of deduplication.

4. RESULTS

4.1 Created corpora

During the development of the crawler we downloaded a total of ca. 4 TB Czech web pages in several web crawler runs. This amounts to ca. 5 billion tokens after all post-processing steps, including deduplication with onion. We merged the corpus with a ca. 2 billion word Czech web corpus we have collected previously by using Heritrix. Since the two corpora overlapped to a high extent, the size of the final Czech web corpus after deduplication is 5.8 billion tokens.

As a next exercise we ran SpiderLing on Tajik Persian to find out how the crawler deals with scarce online resources. We started the crawl from 2570 seed URLs (from 475 distinct domains) collected with Corpus Factory [4]. The crawling finished in two days having no more URLs to download from.

[3] `http://code.google.com/p/justext/`

[4] `http://code.google.com/p/onion/`

Table 3: Final corpora sizes obtained using SpiderLing

target language	corpus size [10^9 tokens]	crawling duration [days]
American Spanish	8.7	14.0
Arabic	6.6	14.0
Czech	5.0	24.4
Japanese	11.1	28.0
Russian	20.2	13.4
Tajik Persian	0.034	1.7
Turkish	3.1	7.0

Since then we focused on crawling widely spread languages such as American Spanish, Japanese and Russian. There are many resources in those languages available on the web, which contributed to quick and effortless crawling.
Since the crawler supports constraining by internet top level domain, the American Spanish corpus was downloaded from national domains of 18 Hispanic American countries. Documents from the same country form a subcorpus of the resulting corpus. Such data may be useful for a research studying varieties in the Spanish language spoken in America. It is worth noting that 75 % of documents in the corpus come from three countries with the highest web presence: Argentina, Mexico and Chile.

There is an overview of all corpora recently built with SpiderLing in Table 3.

4.2 Yield rate

By applying yield rate thresholds on domains we managed to reduce downloading data which is of no use for text corpora and increased the overall average yield rate. Fig. 3 contains the same kind of scatterplot as displayed in Fig. 1, this time on the data downloaded by SpiderLing with Czech as a target language. This is a significant improvement over the previous graph. For low-yielding domains only up to 1 MB of data is downloaded and high amounts of data are only retrieved from high-yielding sources. Many points (i.e. domains) are aligned along the line representing a yield rate of 10 %. Furthermore, the crawling was stopped already at the 512 kB threshold in case of many bad domains.

Note that the graph in Fig. 3 does not take deduplication by onion into account. It displays the size of the data as output by the crawler (i.e. boilerplate removed by jusText, no exactly same documents), not the final deduplicated texts size. Even though the achieved improvement over the previous is indisputable.

We were also interested in the development of the crawling efficiency during crawling. We expected the yield rate to slightly increase over time (the more data downloaded the higher yielding domains selected). The results are pictured by Fig. 4.

Contrary to our expectations, the measured efficiency grew only slightly or stagnated in most cases. We still consider this a good result because even the stagnating yield rates

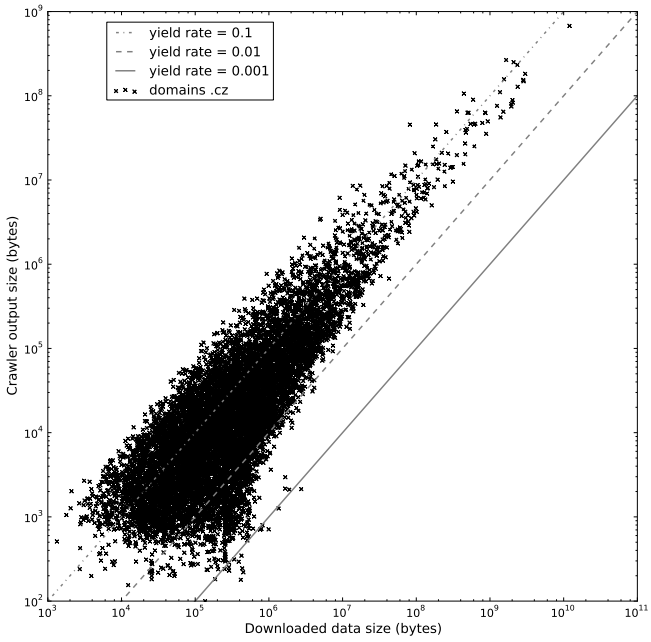

Figure 3: Web domains yield rate for a SpiderLing crawl on the Czech web

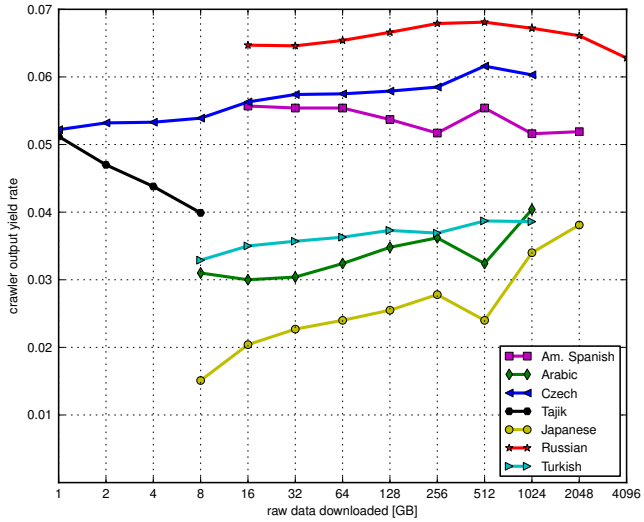

Figure 4: Web domains yield rate for SpiderLing crawls on six target languages

Table 4: Results of crawling the web for large text corpora using SpiderLing

target language	raw data size [GB]	crawler output size [GB]	crawler output yield rate	final corpus size [GB]	final yield rate
Am. Spanish	1874	97.3	0.0519	44.14	0.0236
Arabic	2015	84.4	0.0419	58.04	0.0288
Japanese	2806	110.1	0.0392	61.36	0.0219
Russian	4142	260.1	0.0628	197.5	0.0477
Turkish	1298	51.4	0.0396	19.52	0.0150

were good (with regard to Table 1).
Crawling Japanese was an exception, since the rate kept increasing almost all the time there. The reason may be the starting rate was low. The inbuilt language dependent models (character trigrams, wordlist) may not be adapted well for Japanese and throw away good documents as well.
The less web resources in the target language, the sooner the yield rate drops down. It can be demostrated by the example of Tajik.

The initial yield rate obviously depends on the quality of the seed (initial) URLs. (For example many URLs of electronic newspaper articles in the target language give good initial yield rate.) Irrespective of the seed URLs, the measurements show that sooner or later, the program discovers enough URLs to be able to select good quality domains.

Unlike other languages, crawling Arabic, Japanese and Turkish was not restricted to the respective national domains. That inevitably led to downloading more data in other languages thus throwing away more documents. Considering crawling efficiency in these cases on Fig. 4, the yield rate also depends on constraining crawling to national top level domains.

The yield rate may decrease after downloading a lot of data (the amount depends on the web presence of the target language). In the case of rare languages, the best (text rich) domains get exhausted and the crawler has to select less yielding domains. Concerning decreasing rate while crawling widely spread languages (like Russian), the reason may lie in the design of the crawler. It may be obtaining data from many domains concurrently (ca. 1000–2000 in case of Russian), leaving potentially rich domains waiting and not discovering their true potential.

The final data sizes and average yield rates obtained by crawling five large languages are summarized in Table 4. The final yield rate varies between 1.50 % (Turkish) and 4.77 % (Russian) which is a great improvement over a yield rate of 0.38 % achieved by Heritrix (crawling Portuguese) and a good result compared to a hypothetical yield rate of 1.28 % discussed in section 2.

5. FUTURE WORK

We plan building more huge corpora covering all major languages (French and English being next on the list). Since there are many online resources in these languages, we expect to gather two at least 20 billion tokens corpora in less than a month.
We also need to invest more effort into optimizing the program design to gain more data from scarce resources.
We would like to interconnect the crawler with other linguistic and data processing tools to form a single system offering instant web corpora on demand.

Other plans for the future include analyzing the topics and genres of the downloaded texts and eventually balancing the downloaded content in this respect.

6. CONCLUSION

We presented a way of selective crawling to make obtaining internet content for text corpora more efficient. We have implemented the idea in a new web crawler, which can effectively avoid data not suitable for text corpora thus significantly improving the yield rate of the downloaded documents.

The crawler has already been successfully applied for building billions of words scale corpora in six languages. Texts in the Russian corpus, consisting of 20.2 billions tokens, were downloaded in just 13 days.

Acknowledgements

The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement no. 248307 (PRESEMT project).

7. REFERENCES

[1] M. Baroni, S. Bernardini, A. Ferraresi, and E. Zanchetta. The wacky wide web: A collection of very large linguistically processed web-crawled corpora. *Language Resources and Evaluation*, 43(3):209–226, 2009.

[2] M. Baroni and A. Kilgarriff. Large linguistically-processed web corpora for multiple languages. *Proceedings of European ACL*, 2006.

[3] A. Ferraresi, E. Zanchetta, M. Baroni, and S. Bernardini. Introducing and evaluating ukWaC, a very large web-derived corpus of English. In *Proceedings of the 4th Web as Corpus Workshop (LREC 2008)*, 2008.

[4] A. Kilgarriff, S. Reddy, J. Pomikálek, and A. PVS. A corpus factory for many languages. *Proc. LREC, Malta*, 2010.

[5] J. Pomikálek. *Removing Boilerplate and Duplicate Content from Web Corpora*. PhD thesis, Masaryk University, Brno, 2011.

[6] J. Pomikálek, P. Rychlỳ, and A. Kilgarriff. Scaling to billion-plus word corpora. *Advances in Computational Linguistics*, 41:3–13, 2009.