Top-Level Domain Crawling for Producing Comprehensive Monolingual Corpora from the Web

Dirk Goldhahn[1], Steffen Remus[2], Uwe Quasthoff[1], Chris Biemann[2]
[1]University of Leipzig, Leipzig, Germany
[2]Technische Universität Darmstadt, Darmstadt, Germany
{dgoldhahn, quasthoff}@informatik.uni-leipzig.de, {remus, biem}@cs.tu-darmstadt.de

Abstract

This paper describes crawling and corpus processing in a distributed framework. We present new tools that build upon existing tools like Heritrix and Hadoop. Further, we propose a general workflow for harvesting, cleaning and processing web data from entire top-level domains in order to produce high-quality monolingual corpora using the least amount of language-specific data. We demonstrate the utility of the infrastructure by producing corpora for two under-resourced languages. Web corpus production for targeted languages and/or domains thus becomes feasible for anyone.

Keywords: corpus creation, web crawling, map reduce, web-scale corpora

1. Introduction

With the extraordinary growth of information in the World Wide Web, online documents increasingly become the major source for creating high quality corpora. Unfortunately, the development of technologies that make the information conveniently available to the user causes the process of crawling language data to become even harder. That is why researchers more and more rely on data provided by companies specialized in crawling the web, with all limitations that go along with this (cf. [7]).

We present an approach for creating corpora from the web with only little effort and by using only freely available, open-source software. All components used for data processing can be executed in a highly distributed environment, resulting in quick processing times. Researchers, data analysts and others are hence able to create large-scale high quality corpora targeted towards their own needs. In a case study, we will create two corpora for under-resourced languages, Kiswahili and Faroese. We discuss potential pitfalls, and ways to avoid them.

While there has been a number of initiatives in the past to obtain very large monolingual web corpora, for example WaCky[1] [1], COW[2] [11], Leipzig Corpora Collection [10], or even the very comprehensive *common crawl*[3] provided by Amazon, our contribution lies a) in the comprehensiveness for low-resource languages reached with minimal effort by crawling entire top-level domains, b) in the generic distributed processing pipeline for arbitrary automatic annotations and c) in the availability of the entire processing chain as open-source software component – partially provided by us.

The article is structured as follows: Section 2 describes the proposed approach to crawling and pre-filtering of entire top-level domains. Section 3 presents a generic distributed processing pipeline, which allows us to process very large amounts of data, and Section 4 gives detailed information regarding the availability of the presented tools and the gathered data during the case study described in Section 5. Section 6 summarizes and concludes this work.

2. Crawling

For crawling, we rely on the *Heritrix* project[4] (Version 3.1.1). The Heritrix archival crawler project is an open-source, web-scale crawler made available by the Internet Archive Community. It is used e.g. for periodically creating snapshots of large amounts of webpages in the web, which corresponds to the scheme of creating corpora from the web. *Heritrix* is a versatile tool, providing many options to configure the desired crawling behavior. Compared with other crawling software like *wget*, *HTTrack*, or *Nutch*, it offers several general advantages: Single crawl jobs can cover hundreds of millions of pages; it is stable, fast and follows more links than other comparable tools due to better handling of Java-script links while it is still easy to use.

Heritrix is initialized with a list of specified webpages – called seed – from which it extracts web links to other webpages that are subsequently downloaded and processed accordingly. Here, we will use it to harvest entire Top Level Domains (TLD), which means we download every suited web document we encounter in a particular TLD. The initially provided list of up to 2,000 seed domains for each TLD contains randomly chosen URLs coming from previous crawls. The composition of the seed has only minor influence on the results of the crawling process: Typically, hubs of a TLD – i.e. websites that contain links to a many different websites – are reached within the first steps of the process. We configured Heritrix to extract links from URLs and to follow them while not leaving the current TLD and not downloading the same URL twice or more.

[1] http://wacky.sslmit.unibo.it/
[2] http://hpsg.fu-berlin.de/cow/
[3] http://commoncrawl.org/
[4] http://crawler.archive.org

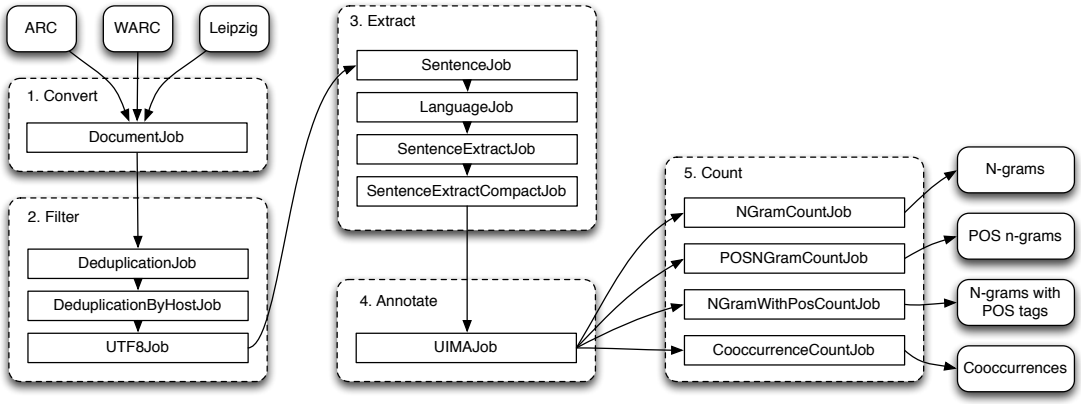

Figure 1: The individual jobs in the standard WebCorpus pipeline. Figure taken from [3].

Running on a machine with 8 CPU-cores and 20GB of RAM using an unthrottled 1GBit/s Internet connection, it reaches crawling speeds of up to 200 URLs/s for each crawling job while running up to 3 jobs in parallel. We chose to reduce load for single servers by limiting queries to the same domain to one per seven seconds. Hence, high crawling speed is only achieved as long as many servers are queued. To increase crawling performance, some basic configurations were considered: In order to avoid link farms and spider traps, we follow links only up to a maximum depth. To reduce download bandwidth we exclude certain kinds of files like images, media files, compressed archives or executable files. Additionally, URLs containing certain keywords (*download*, *files*, *image*, *pics*, *upload*, *redir* or *search*) are excluded from consideration. Further, we restrict the maximum file size to 1 MB to reduce the amount of lists or computer-generated content.

Heritrix creates output files in the Web Archive file format (WARC)[5]. The WARC file format specifies how to combine multiple digital resources with meta-information into a single file for long term archiving or distribution. Further processing steps proposed in this work operate on this representation.

3. Processing and Analyzing Web-Data

Post-processing of harvested web data can be efficiently performed using the *WebCorpus*[6] project. WebCorpus makes use of the highly efficient *Hadoop*[7] framework, which offers the execution of algorithms following the *MapReduce* programming paradigm [6] in a distributed environment. Due to the choice of Hadoop as the basis framework it is possible to process very large data in parallel by a number of computers or just by a single machine. The core idea in MapReduce is to split an algorithm into two phases: *map* and *reduce*. In the map phase, so-called key-value pairs of the input data are produced which are subsequently grouped and combined in the reduce phase by their key to produce the final result. In terms of Hadoop, an algorithm following the MapReduce programming principle is called a *HadoopJob*. The WebCorpus project provides individual HadoopJobs, which are designed to process the data in a pipeline fashion, i.e. one HadoopJob after another.

The general steps of the processing pipeline are described by:

1. **Convert**: converting input data – currently supported input file formats are WARC, ARC[8] and Leipzig Corpora Collection [10] – into a unified document representation, thereby optionally removing html boilerplate text (cf. e.g. [8]),
2. **Filter:** removing duplicate, broken or pointless documents,
3. **Extract:** segmenting, filtering and merging texts in the desired level of granularity – e.g. unique sentences, paragraphs or documents in a particular language,
4. **Annotate:** process texts with UIMA [9] components, e.g. tokenizing, tagging, etc., and parsing
5. **Count:** exploit the resulting annotations by counting n-grams, co-occurrences, subtrees of dependency parses, etc. in the annotated texts.

Figure 1 shows a more detailed overview of the different HadoopJobs in the respective phases. For a description of the individual jobs the reader is referred to [3].

Some of the jobs, in particular the *LanguageJob* and partially also the *SentenceJob* and the *UIMAJob*, are language dependent. For example, the LanguageJob uses the language identification package (JLanI) from the ASV-Toolbox[10] [4], which relies on a precomputed list of high-frequency words for a particular language. These word lists are available for more than 500 languages using mainly Wikipedias, the Universal Declaration of Human Rights and religious texts. More languages could

[5] http://www.digitalpreservation.gov/formats/fdd/fdd000236.shtml

[6] http://sf.net/projects/webcorpus

[7] http://hadoop.apache.org

[8] Internet Archive file format http://www.digitalpreservation.gov/formats/fdd/fdd000235.shtml

[9] http://uima.apache.org

[10] http://wortschatz.uni-leipzig.de/~cbiemann/software/toolbox/

be included on the basis of Bible texts (cf. [5]). Likewise, the *SentenceJob*[11] uses handcrafted sentence breaking rules to segment sentences. While it is equipped with a generic rule set, more specific rules for a particular language will certainly improve the results, but were not considered for the case study in this work for the sake of generality.

Processing 10 GB of web data took around one hour on our compute cluster consisting of eight nodes with eight cores each. The runtime complexity solely depends on the amount of input data and the number of provided machines [1].

The cleaned, language-filtered and preprocessed documents, as well as the various outputs of the count phases like statistically significant co-occurrences or n-grams can then be exploited by a variety of applications, e.g. distributional thesauri or language models (cf. e.g. [2]). In this work, we will exemplify the data with visual analysis of significant co-occurrences using *CoocViewer* [12] [9]. With CoocViewer, co-occurrences of words from multiple arbitrary text corpora can be explored visually in a user-friendly way, providing also access to the source text via full-text indexing. The application itself is divided into two major components:

1. the server-sided data management part, where data is stored in a relational database for fast access through indexes (cf. [10]), and
2. the web based front end, which runs on top of an http server[13]. The browser based client application is thus independent of the underlying operating system and available for many users accordingly.

Screenshots of the application follow in Section 4 where we show the feasibility of processing web-data based on two sample web crawls. As a key characteristic, CoocViewer also comes with the possibility to visualize significant concordances. This feature is particularly useful for analyzing high frequency words.

4. Availability

Corpora as described in the following case study are made available in various ways. On the one hand the full size corpora are accessible online using the web interface of the Leipzig Corpora Collection[14]. On the other hand the textual data can be downloaded[15]. For download corpora of standard sizes of up to 1 million sentences are provided. They can be viewed locally using e.g. the Java Corpus Browser [10]. All textual data underlies creative commons attribution license (cc by)[16] allowing users to use and modify the data freely.

The collected sentences are shuffled such that the original structure of the documents cannot be recovered easily, because of legal issues. This inhibits the reconstruction of the original material. With respect to German copyright legislation this practice is considered legally secured, since there is no copyright on single sentences.

The various pre-processing tools involved in the creation of corpora as described are free to use. Among these are tools for HTML-Stripping, sentence segmentation, sentence cleaning, and language identification[17]. All tools can be utilized for non-commercial users following creative commons attribution-noncommercial license (cc by-nc)[18]. The WebCorpus and the CoocViewer toolkits are available as open-source components in Java under the Apache v2 License[19].

5. Case Study

For our case study, two top-level-domains were crawled, from which we assume that they contain documents of languages that are known to be under-resourced. We tested the .fo domain (Faeroe Islands) and the .ke domain (Kenya), where the languages of interest are Faroese and Kiswahili respectively. Kiswahili is also spoken in other countries such as Tanzania, which could be collected by crawling their respective TLDs. Both domains were crawled using the Heritrix-based crawler, resulting in 1.2 million websites for .fo and 3.1 million websites for .ke. Crawling took about three days for Faroe Islands and four days for Kenya resulting in an average speed of 9 resp. 5 URLs per second. Due to self-imposed politeness restrictions, a maximum download speed of about 200 URLs/s was only reached at the beginning of the crawling process. Higher average rates could easily be achieved by lifting query limits for the cost of being less polite to web server operators.

When conducting language separation, fundamentally different compositions of the domains in question become obvious. More than 60% of the documents of the .fo TLD are written in Faroese, as can be seen in Table 1. English is the second largest language having a 15% share. Next in the ranking are further North Germanic languages, namely Icelandic and Danish.

Table1: Results of language separation using websites of the Faroese domain.

Language	Percentage
Faroese	60.63
English	14.69
Icelandic	11.24
Danish	10.29
French	0.64

[11] The SentenceJob internally uses the ASV-Toolbox.
[12] http://coocviewer.sf.net
[13] Any webserver that supports PHP.
[14] http://corpora.informatik.uni-leipzig.de
[15] http://corpora.informatik.uni-leipzig.de/download.html
[16] https://creativecommons.org/licenses/by/3.0/
[17] http://asvdoku.informatik.uni-leipzig.de/corpora/index.php?id=corpus-pre-processing-tools
[18] http://creativecommons.org/licenses/by-nc/3.0/
[19] http://www.apache.org/licenses/LICENSE-2.0

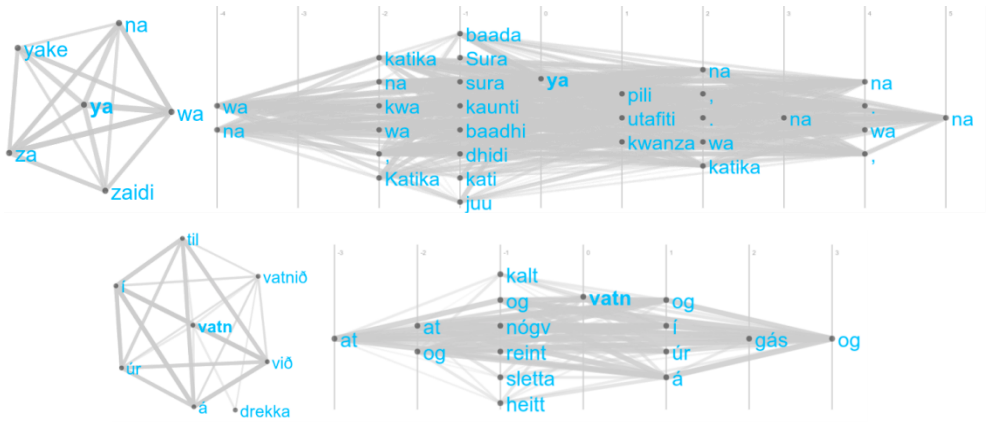

Figure 2: Significant co-occurrences and concordances with the word 'ya' (engl. pendant is roughly 'of', frequency 10,531) from the Kiswahili corpus (upper) and with the word 'vatn' (engl. 'water', frequency 1,504) from the Faroese corpus (lower). Thicker lines represent more significant relations.

When analyzing the Kenyan TLD, an extremely high percentage value for English documents becomes evident (Table 2). Although it is the second largest language among .ke documents, only 0.84% of all texts contain Kiswahili. Together, these two form the national languages of Kenya.

Table2: Results of language separation using websites of the Kenyan domain.

Language	Percentage
English	98.20
Kiswahili	0.84
Russian	0.21
Latin	0.12
Spanish	0.08

The WebCorpus framework was applied as described in Section 3. Only Faroese respectively Kiswahili texts were considered, texts from other languages were filtered in the Filter phase of the WebCorpus pipeline (cf. Sec. 3). Further, we defined our unit of granularity to be the sentence level since our example application is the analysis of co-occurrences of words on a sentence level. After applying the entire WebCorpus pipeline as described in Section 3, we have 7,873 unique sentences and 31,274 types for Kiswahili, and 888,255 unique sentences and 1,030,611 types for Faroese. Yet, the lack of a profound knowledge of these languages makes it impossible for us to judge the quality of the extracted sentences. In particular, the very high number of sentences and tokens for Faroese suggests unclear boundaries in the language separation step. Indeed, during manual inspection of the dataset, we observed some false-positive Danish sentences.

Figure 2 shows the co-occurrences and significant concordances of selected words from either corpus. As should become evident, the setup we described is suited for studies in corpus linguistics and other research.

6. Conclusion

Free and open-source software components have been made available by us and others, that allow researchers and others to produce high quality web corpora targeted to their own needs without relying on the good will of commercial companies to provide it. We have exercised one possible workflow for producing such corpora for two under-resourced languages and conclude, that although we are lacking the needed knowledge for these languages, we are able to produce reasonable results. We assume that further processing of these corpora by experts – mainly cleaning of artefacts from different languages, false segmentation, etc. – would result in high quality corpora from the web. Everybody is thus able to produce web corpora using just the few steps outlined above, and by relying solely on freely available software. By applying some simple configuration settings for Heritrix, the open-source crawler of the Internet Archive, it is easy to crawl specified regions of the World Wide Web in order to collect usable text. By making use of the Hadoop framework, the user herself chooses the level of scalability. Even a single computer is able to run the provided workflow, but when providing more machines, users are able to create corpora of very large sizes in reasonable time.

In two case studies, we have demonstrated how to collect corpora for rather under-resourced languages. Still, the proposed approach can be applied to larger languages if enough computational resources are available. These corpora can form the basis to compute language models, and other NLP components trained from unannotated text.

7. References

[1] 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.

[2] Thorsten Brants, Alex Franz (2006): Web 1T 5-gram

Version 1. *LDC*, Philadelphia, PA, USA

[3] Chris Biemann, Felix Bildhauer, Stefan Evert, Dirk Goldhahn, Uwe Quasthoff, Roland Schäfer, Johannes Simon, Leonard Swiezinski, Torsten Zesch. 2013. Scalable Construction of High-Quality Web Corpora. *Journal for Language Technology and Computational Linguistics (JLCL)*, 28(2), pp. 23–59.

[4] Chris Biemann, Uwe Quasthoff, Gerhard Heyer, and Florian Holz. 2008. ASV Toolbox – A Modular Collection of Language Exploration Tools. In *Proc. of LREC 2008*, Marrakech, Marocco

[5] Michael Cysouw. 2009. Using the World Atlas of Language Structures. In *STUF - Language Typology and Universals Sprachtypologie und Universalienforschung.* 61(3): 181–185. Akademie Verlag, Berlin

[6] Jeffrey Dean and Sanjay Ghemawat. 2004. MapReduce: Simplified Data Processing on Large Clusters. In *Proc. of the 6th Symposium on OSDI*. San Francicso, CA, USA

[7] Adam Kilgarriff. 2007. Googleology is Bad Science. *Computational Linguistics.* 33(1): 147–151.

[8] Christian Kohlschütter, Peter Fankhauser and Wolfgang Nejdl. 2010. Boilerplate Detection using Shallow Text Features. In *Proc. of WSDM,* New York City, NY, USA

[9] Janneke Rauscher, Leonhard Swiezinski, Martin Riedl and Chris Biemann. 2013. Exploring Cities in Crime: Significant Concordance and Co-occurrence in Quantitative Literary Analysis. In *Proc. of the Computational Linguistics for Literature Workshop at NAACL-HLT 2013*, Atlanta, GA, USA

[10] Matthias Richter, Uwe Quasthoff, Erla Hallsteinsdóttir, and Chris Biemann. 2006. Exploiting the Leipzig corpora collection. In *Proc. of the IS-LTC.* Ljubljana, Slovenia.

[11] Roland Schäfer and Felix Bildhauer. 2012. Building large corpora from the web using a new efficient tool chain. In *Proc. of LREC 2012*, Istanbul, Turkey