LanguageCrawl: A Generic Tool for Building Language Models Upon Common-Crawl

Szymon Roziewski, Wojciech Stokowiec

National Information Processing Institute, Warsaw, Poland

Wait, but why?

The Internet is the largest and most diverse collection of textual information in human history, it covers almost all known subjects and languages. It constitutes an appealing resource for extraction of large-scale corpus for language modelling. However, until recently, it was highly unlikely that language researchers in the academia have had access to the necessary infrastructure needed to process the Internet in an effort to build a language corpus. With recent improvement of computing power, storage availability and powerful, highly efficient scalable processing and computing frameworks, it has become feasible to build a large scale corpus using commodity hardware and publicly available web-archives.

Our tool **LanguageCrawl** enables NLP researchers to easily construct large-scale corpus of a given language filtered directly from Common Crawl Archive. We believe that the linguistic community could benefit from our work: precomputed N-grams or distributed word representations could be used for example to boost the accuracy of machine translation systems.

Common Crawl

Common Crawl Archive, on which our tool has been based, is an open repository of web crawl information containing petabytes of data. It is currently stored in three formats: raw data from crawl (WARC), HTML with extended metadata (WAT) and extracted plaintex with minimal amount of metadata (WET).
As most NLP tasks require only textual information, we have decided to build our tool around WET files. Our use-cases have been based on corpora extracted from January Crawl Archive, which is approximately 140 TB in size and contains 1.82 billion web pages.

Common Crawl Archive: `http://commoncrawl.org`

Akka framework: `http://akka.io/`

Our Website: `http://commoncrawl.org/`

Actor Model

In order to facilitate data processing, Common Crawl Foundation have divided each of it's crawl archives into several 140 MB gz-compressed files. Since textual information resides in disjoint files, it is straightforward to process data in parallel. Due to the fact, that processing web-scale data requires not only passing millions of messages concurrently, but also handling multiple failures (e.g. data store unavailability or network failures), we have decided to use Akka framework: a high-performing, resilient, actor-based runtime for managing concurrency.

In our application, actor system creates Master Actor (M), an actor responsible for iterating over file that list WET files and dispatching their URL paths to individual File Workers. In an effort to avoid context-switching we have decided to limit the number of File Workers (w_i, $i = 1, \ldots, N$) to the number of cores available in the cluster on which program has been run. After receiving message with URL address each File Worker begins downloading compressed file and decompresses incoming stream of bytes simultaneously processing WET files line-by-line, in an effort to extract individual pages. Subsequently, each of the extracted pages is sent to Bouncer Actor (b_j, $j = 1, \ldots, M$), which filters pages according to language preferences with the Chromium Compact Language Detector 2. Finally, after successful language verification, website text is stored in Cassandra data base (Db). The flow of messages is shown in figure 1.

Actor Communication

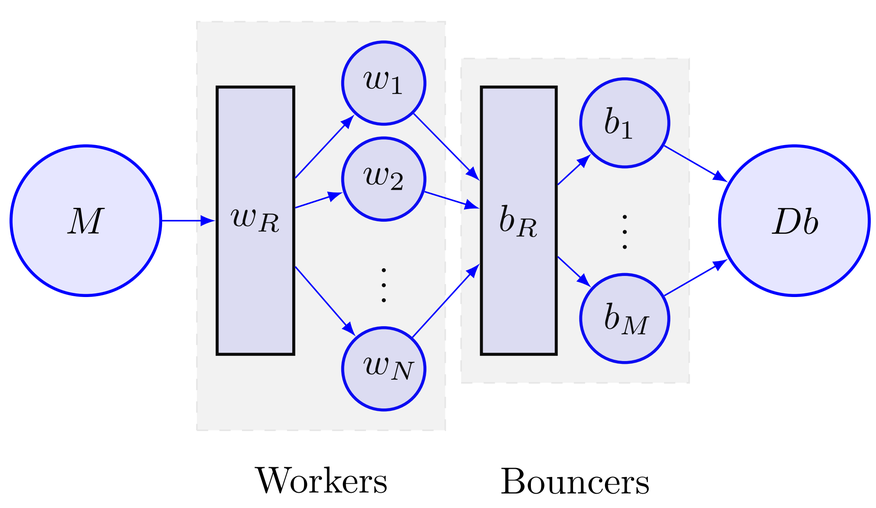

Fig. 1: Message flow in our actor system

N-gram occurrences by type

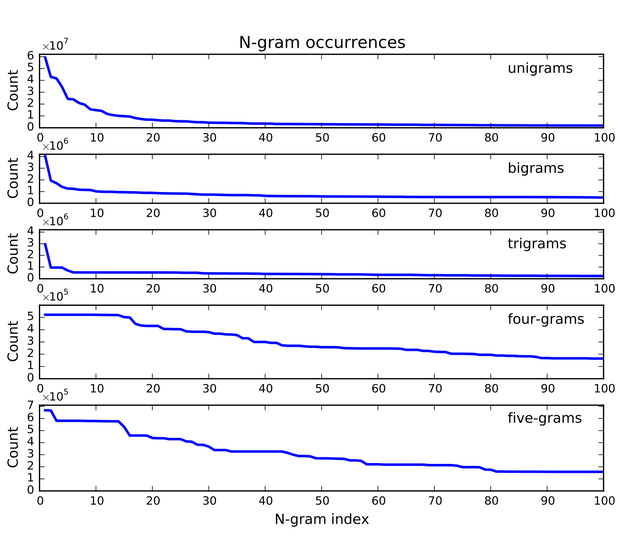

Fig. 2: N-gram occurrences by type

This chart summarizes occurrences each of N-gram type for the most frequent 100 N-grams. The line charts show changes in N-gram counts from the N-gram frequency table. The shapes of N-gram occurrence functions are more steep for unigrams, bigrams and trigrams, whereas for four- and five-grams the curvature is more flat, with long tails widening to the end of the N-gram collection.

N-gram counts by type

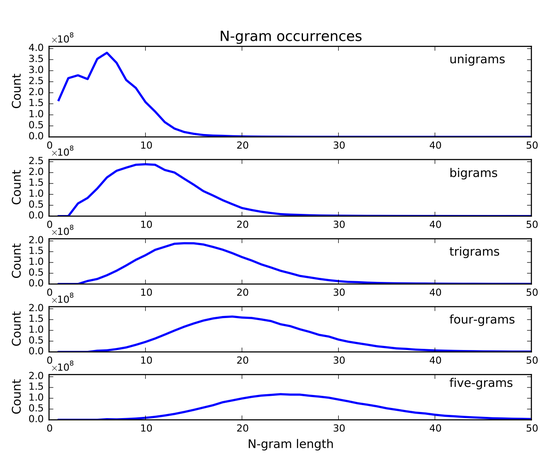

Fig. 3: N-gram counts by type

The N-gram curves show counts of each N-gram type with respect to its length by the means of characters, and illustrate the data from table 4. The line chart for unigrams shows two maxima, the first one is related to stop words and the second one is for the most frequent size of Polish word which is 6.

N-gram statistics

N-gram Type	Total # occurrences	Collection Size
unigram	2,985,800	50 MB
bigram	2,608,100	60 MB
trigram	3,790,700	113 MB
four-gram	4,277,000	159 MB
five-gram	3,617,000	163 MB
total	17,278,600	545 MB

Fig. 4: Overview of total numbers of unique N-grams after cleaning

N-gram Type	Mean	SE	Median	10th	90th
unigram	8.79	3.07	9	5	13
bigram	15.35	4.61	15	10	21
trigram	22.32	6.03	22	15	30
four-gram	30.03	7.33	29	21	39
five-gram	38.34	8.46	38	28	49

Fig. 5: Statistics of unique N-grams' lengths