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

Szymon Roziewski, Wojciech Stokowiec
National Information Processing Institute
Natural Language Processing Laboratory, Warsaw, Poland
sroziewski@opi.org.pl, wstokowiec@opi.org.pl

Abstract

The web data contains immense amount of data, hundreds of billion words are waiting to be extracted and used for language research. In this work we introduce our tool **LanguageCrawl** which allows Natural Language Processing (NLP) researchers to easily construct web-scale corpus the from Common Crawl Archive: a petabyte scale open repository of web crawl information. Three use-cases are presented: filtering Polish websites, building N-gram corpora and training continuous skip-gram language model with hierarchical softmax. Each of them has been implemented within the LanguageCrawl toolkit, with the possibility to adjust specified language and N-gram ranks. Special effort has been put on high computing efficiency, by applying highly concurrent multitasking. We make our tool publicly available to enrich NLP resources. We strongly believe that our work will help to facilitate NLP research, especially in under-resourced languages, where the lack of appropriately sized corpora is a serious hindrance to applying data-intensive methods, such as deep neural networks.

Keywords: Polish Web Corpus, Common Crawl, Word2Vec

1. Introduction

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 corpora for language modelling. However, until recently, it was highly unlikely that language researchers in the academia would have had access to the necessary infrastructure needed to process the Internet in an effort to build a language corpus. With the recent improvements 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[1] and publicly available web-archives.

Our tool **LanguageCrawl** enables NLP researchers to easily construct large-scale corpus of a given language filtered directly from the 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 (Buck et al., 2014).

In this work our primary objective is to illustrate the use cases for LanguageCrawl: filtering Polish Websites from the Common Crawl Archive and subsequently building N-gram corpora and training continuous Skip-gram language model. As far as we know, nobody has formed out such a collection made from the Polish Internet. We are interested in Polish language since we would like to use our results for further NLP research for Polish, and to enrich general knowledge about it, i.e. models and data. Both, source code and the language models will be made publicly available by the time this paper is published.

[1]By this term we understand a piece of fairly standard hardware that can be purchased at retail, such as low-cost desktop computers.

1.1. Data Set

Common Crawl Archive[2], on which our tool has been based, is an open repository of web crawl information containing petabytes of data.

As most NLP tasks require only textual information, we have decided to build our tool around WET files containing plaintext with minimal amount of metadata. 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.

Processing data from the Common Crawl Archive is a massive task, very limited to the Internet bandwidth connection, which is a severe bottleneck for data fetching. In our case it took weeks to download enough data to build a reasonable Polish website corpora. Our LanguageCrawl toolkit provides highly concurrent actor-based architecture for building local Common Crawl Archive and N-gram collection for a specified language. Textual data processing, cleaning and N-gram building are accomplished within the same aforementioned, effective resource undemanding system. We have used 36 actors[3], which are run on an efficient computing machine, to perform these tasks.

1.2. 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

[2]http://commoncrawl.org/

[3]Higher level of abstraction objects, crucial parts of Actor Model implementation.

framework[4] : a high-performing, resilient, actor-based runtime for managing concurrency.
Actors are objects which encapsulate state and behaviour and communicate with each other by exchanging messages. Theoretical underpinnings for actor model, a mathematical model of concurrent computation, are described in (Hewitt et al., 1973). In Akka, actors are organized in hierarchical structure known as actor system which is equipped with supervisor strategy, i.e. a straightforward mechanism for defining fault-tolerant service.
In our application, ActorSystem creates FileMaster, an actor responsible for iterating over list of WET files and dispatching their URL paths to individual FileWorkers. In an effort to avoid context-switching we have decided to limit the number of FileWorkers to the number of cores available in the cluster on which program has been run.
In Figure 1 a message passing diagram is shown. ActorSystem begins the processing flow by sending a message to FileMaster. Afterwards, FileMaster processes URLs linking to the data in Common Crawl Archive, creates its FileWorkers (workers) and feeds them with one URL until all links are fetched. In the mean while, workers process given URL by downloading data and extracting textual content. Subsequently, they recognize individual pages and send a specimen of each for language recognition to available Bouncer Actors.
The language detection module uses a very efficient and accurate package, optimized for space and speed, which is Compact Language Detector 2 (CLD2) package[5]. It is based on a probabilistic model – a Naive Bayes classifier. As the input for CLD2, first one hundred words from the current document, are used. The CLD2 package has several embellishments for the algorithm improvement. Web-specific words containing no language information has been truncated: page, click, link, copyright, etc. Repetitive words that could affect the scoring like jpg, foo.png, bar.gif, have been stripped away.
After content written in predefined language has been detected, Bouncers perform simple spelling correction[6] on each word from the text and subsequently write the corrected content to the Cassandra Database.

2. Related Work

There are a few studies on using Common Crawl Data for N-gram generation, which corresponds to concepts and entities in NLP. One of them is presented in the paper (Kanerva et al., 2014), which gives an overview on possible applications of Common Crawl Data. They have obtained both linear and syntactic N-gram Collection from a Finnish Internet Crawl, and made them publicly available. Some more technical issues have been demonstrated, specifically

[4]http://akka.io

[5]Language detection is conducted with the Chromium Compact Language Detector 2

[6]We correct words with edit distance equal or smaller than 2. Vocabulary of correct words is build on subset of literary texts taken from *National Corpus of Polish* (Przepiórkowski et al., 2008), (Institute of Computer Science at the Polish Academy of Sciences, 2010).

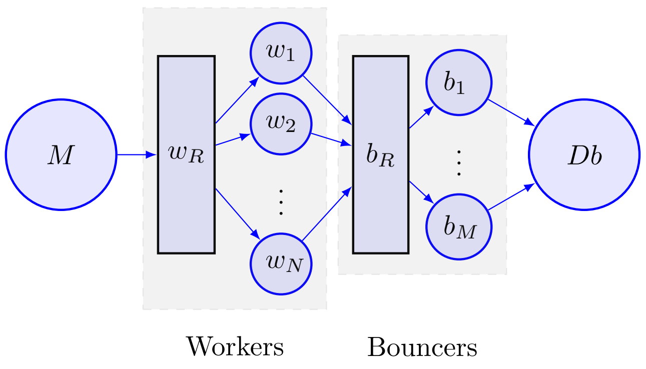

Figure 1: Message passing diagram. FileMaster (M) sends message to its FileWorkers ($w_1, \ldots, w_N$) wrapped in the router w_R. FileWorkers needs Bouncers ($b_1, \ldots, b_M$) as helpers for language detection and writing data to Cassandra Storage. b_R is a router for Bouncer Actors. Db is Cassandra Storage.

raw textual data processing and language detection. Another study (Buck et al., 2014) concerning N-gram counts and language models formed on Common Crawl Archive is broader by the number of languages analysed. In this study, the authors stress the importance of data duplication and normalization. Additionally, they compare perplexity of N-gram language models trained on corpora allowed by the constrained condition of the 2014 Workshop on Statistical Machine Translation and report that the lowest perplexity was achieved on their model. Moreover, the authors report that adding the language models presented in their work to a Machine Translation (MT) system improves BLEU score between 0.5 and 1.5 percentage points.
In (Ziółko et al., 2010) a few Polish corpora have been studied, statistics for N-grams have been computed, and a tool for manual N-gram error correction have been proposed. The authors pre-processed textual data more carefully with respect to Polish diacritical marks. They also used morphological analyser for better word recognition. Obtained N-gram model is used to automatic speech recognition.
The authors of paper (Ginter and Kanerva, 2014) train word2vec skip-gram model on 5-grams both from the Finnish Corpus extracted from the Common Crawl Internet crawl dataset, and Google-Books n-gram collection. Among others, they concern word similarity and translation tasks. In the first task semantic and syntactic queries are used to recover word similarities. The second task itself is about testing the ability of word2vec representation to simple linear transformation from one language vector space to another. Training word2vec models on N-gram collection has the advantage of its compactness. Speedup for English N-gram corpus is increased nearly 400 times, what means it is possible to do the computations even on a single machine.
Unlike the above approaches, we are more focused on Polish language and present the following:

- LanguageCrawl (our NLP toolkit),

- Polish N-gram collection,

- word2vec model trained on Polish Internet Crawl based on Common Crawl Archive.

We provide more statistical information about Polish language and show N-gram counts and their distribution, which is built on the basis of the huge Polish Internet Crawl, much greater in size than corpora analysed in (Ziółko et al., 2010). Additionally, this study improves on previous works by using actor model, which enables us to take advantage of all available cores by highly parallelizing the corpus building process. As a result, LanguageCrawl scales well with the increase in the number of worker nodes.

3. Use Cases

3.1. N-gram Language Model

Based on Common Crawl Data that we have scraped and processed, an N-gram model has been constructed.

After sorting out non-Polish content websites, we preprocessed the data by: converting to lowercase letters, splitting into sentences by using tokenizer from NLTK Python toolkit (Loper and Bird, 2002), removing non-alphanumerical characters, truncating fused and misprinted words whose lengths were relatively long and were beyond the spell-corrector's skills. Polish diacritical marks and stop words have been preserved, because they are useful for phrase search (Ziółko et al., 2010). After creating the N-gram model, duplicated N-grams have been removed. As a result, data set size has been reduced by approximately 99.4%. In the same time, removing fused and misprinted words reduced data volume by 63%. This indicates, that raw common crawl data are highly duplicated. It is therefore worth investigating how early deduplication, e.g. removing copyright notices, influences resulting corpora. Our Polish N-gram corpora are formed out of five N-gram types, starting from unigrams and ending on five-grams.

Table 1 summarizes the corpora by the means of N-grams occurrences and their sizes given in MB with respect to N-gram type. Our Polish N-gram collection extends above 17M entries. Nonetheless, this data set could have been made much larger, simply by processing additional crawl archives.

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

Table 1: Overview of the total numbers of unique N-grams and their weights after truncating of misprints and fused words

A survey of the length of the different N-grams might be interesting for language researchers. We offer additional insights on the topic, beyond what can be found in the typical dictionary, thanks to the bigger size of the corpus we analysed, the statistical measures we provided and the unique textual data source we gathered from websites. We believe that all of that may enrich Polish language characteristics and shed new light on its analysis.

Table 2 presents several statistics of the N-gram length for each N-gram type i.e. mean, standard error, median, and two percentiles: 10th and 90th. It has been inferred that the average Polish word is 8 letters long, what seems to be fairly overestimated due to the fact that a number of fused words are still undetected and remain in the N-gram corpora. Thus all of the statistics may be affected with this issue more or less.

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

Table 2: Statistics of unique N-grams' lengths

In Figure 2, we show N-gram occurrences for each N-gram type with respect to its length. A few interesting phenomena concerning Polish N-gram corpora can be seen: the number of occurrences have the same order of magnitude; for higher order N-grams, chart's shapes are more bell-curve-like, broadening with increasing N-gram rank; charts are more right-skewed and becoming flatten with greater N-gram order; the mean and median are shifted to growing N-gram length, what is caused by long right tail of curves. The elongated tails for bigger N-gram length may exist because of undetected words concatenation. The chart showing unigrams distribution is different in particular from the others, having two local maxima. The first maximum may be the result of stop words existence in the N-gram corpora, whereas the second one, should be the most frequent word length in Polish language based on Common Crawl Data, which is 6. Stop words are frequently occurring in the corpora, roughly 44% out of 274 (the number of all Polish stop words) first most recurring unigrams in our corpora are stop words. We have estimated this ratio by comparing both numbers of: the most frequent unique unigrams (only stop words), and Polish stop words, which have been extracted from NLTK Python tool (Loper and Bird, 2002). The average length of those 44% unigrams is 3.32, which is in agreement with the first lower maximum in Figure 2. We can recognize that the maxima for higher-rank N-grams are shifted roughly by 6. We may infer from this fact that the average length of Polish word is about that value, depending on Polish N-gram corpora.

The charts from Figure 3 present the occurrences of the top 100 N-grams for each N-gram rank analysed. One can notice that the Zipf's Law manifests itself by asymptotic decay of N-gram occurrences, especially for the first three curves. The aforementioned law states that the word occurrence in a corpora is inversely proportional to its index in the frequency list. The curves showing four- and five-grams are more flat, which can be caused by the fact that those N-grams are more correlated to each other among the N-grams within a rank.

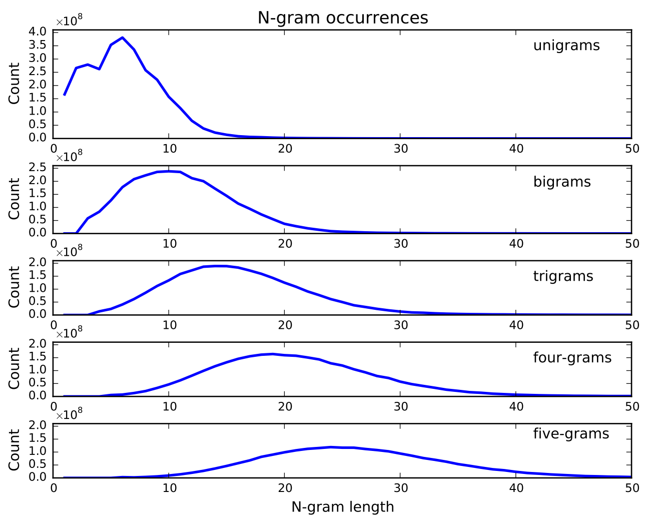

Figure 2: The N-gram curves show counts of each N-gram type with respect to its length by the means of characters, and illustrate the data from Table 1. The line chart for unigrams shows two maxima, the first one is related to stop words and the second one is for the most frequent size of Polish word which is 6. The curves of the N-gram occurrence functions for bigrams, trigrams, four- and five-grams are right-skewed bell-curve-like, with long tails widening with increasing N-gram length. The chart curvatures lessen with increasing N-gram rank. The maxima are right-shifted by the value of 5.

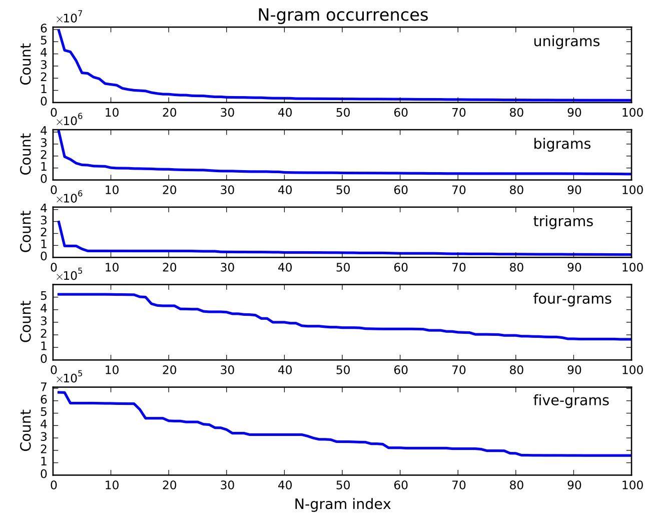

Figure 3: The N-gram charts summarize occurrences each of N-gram type for the most frequent 100 N-grams. The line charts show changes in N-gram counts from the N-gram frequency table. The shapes of N-gram occurrence functions are more steep for unigrams, bigrams and trigrams, whereas for four- and five-grams the curvature is more flat, with long tails widening to the end of the N-gram collection. The general trend is asymptotic downwards for the first three charts and a smooth decay for the others.

3.2. Word2Vec

Word2Vec computes distributional vector representation of words which has been shown to help learning algorithms to boost their scores in natural language processing tasks (Mikolov et al., 2013). Continuous bag-of-words and skip-gram architectures yield vector representations of words, which are useful in various natural language processing applications such as machine translation, named-entity recognition, word sense disambiguation, tagging, parsing etc. Skip-gram model is an idea of learning word vector representations which are useful in predicting its sense in the same sentence. The aim of the skip-gram model is to maximize the function of average log-likelihood, given a sequence of training words $w_1, w_2, \ldots, w_T$:

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{j=-k}^{k}\log p(w_{t+j}|w_t), \tag{1}$$

where k stands for the size of the training window. Each word w corresponds to two vectors u_w and v_w that are vector representations of w as word and sense accordingly. The probability of accurately predicting word w_i given w_j is described by the softmax model

$$p(w_i|w_j) = \frac{\exp(u_{w_i}^\top v_{w_j})}{\sum_{l=1}^{V}\exp(u_l^\top v_{w_j})} \tag{2}$$

where V is the size of vocabulary. The skip-gram model featured with softmax is computational expensive, computing $\log p(w_i|w_j)$ is proportional to vocabulary size, which can reach size of billion. For boosting model efficiency, we use hierarchical softmax with lower computation demands bounded by $O(\log(V))$ as shown in (Mikolov et al., 2013). In our service we have used the Gensim (Řehůřek and Sojka, 2010) implementation of word2vec. Gensim implementation is fast enough to process the filtered corpus in less than one day. Additionally, in an effort to reduce memory usage, our training pipeline takes advantage of iterators. The model is being trained in on-line fashion, by fetching documents one after another from the database. Finally, the resulting model is about 5 GB large, which makes it possible to train it even on machines with modest amount of available RAM.
In Table 3 words with entries closest in meaning are presented. The output is considered to be semantically coherent.

Table 3: Examples of semantic similarities based on word2vec trained on Polish Internet Corpora.

Word	**Most Similar**	**Distance**
król (*king*)	cesarz (*emperor*)	0.73
Tusk (*former PM*)	Donald (*his name*)	0.74
kobieta (*woman*)	dziewczyna (*girl*)	0.80
meżczyzna (*man*)	chłopak (*boy*)	0.85
dziewczyna (*girl*)	rozochocna (*horny*)	0.82
apple	tablety (*tablets*)	0.79
Dublin	Blessington	0.85
Sushi	Pizza	0.76

A few examples of linguistic computations based on vector space representation are shown in Table 4. Thanks to word2vec we have a direct link between the mathematical representation and the semantic meaning of a word.

Table 4: Linguistic regularities in vector space.

Expression	Nearest Token
król − meżczyzna + kobieta (*king − man + woman*)	Edyp (*Oedipus*)
wiekszy − duży + mały (*bigger − big + small*)	mniejszy (*smaller*)
Włochy − Rzym + Francja (*Italy − Rome + France*)	Paryż (*Paris*)
dżungla + król (*jungle + king*)	Tarzan | król lew (*Tarzan*) | (*lionking*)

4. Possible Applications, Conclusions and Future Work

We have demonstrated the abilities of our LanguageCrawl tool, which are mainly scraping Common Crawl Archive with respect to a given language, and both building N-gram model and training word2vec on that data. The N-gram model can be incorporated into machine translation system to boost its performance, as has been shown in (Buck et al., 2014). Researchers may benefit from well-trained word2vec model on large-scale Polish corpora.

Statistics over a large Polish Internet corpora provide interesting insights. In this study, we have shown Polish five N-gram types distributions, estimated the mean length of Polish words, and provided their statistical characteristics.

As a future work, we would like to enrich our toolkit with syntactic N-grams and train word2vec on both linear and syntactic N-gram collections.

5. Acknowledgements

I am personally very grateful to Dr. Irina Temnikova for her precious bright remarks and insights.

6. Bibliographical References

Buck, C., Heafield, K., and van Ooyen, B. (2014). N-gram counts and language models from the common crawl. In *Proceedings of the Language Resources and Evaluation Conference*, Reykjavk, Icelandik, Iceland, May.

Ginter, F. and Kanerva, J. (2014). Fast training of word2vec representations using n-gram corpora. In *Linköping Electronic Conference Proceedings*, Uppsala University, November.

Hewitt, C., Bishop, P., and Steiger, R. (1973). A universal modular actor formalism for artificial intelligence. In *Proceedings of the 3rd International Joint Conference on Artificial Intelligence*, IJCAI'73, pages 235–245, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Kanerva, J., Luotolahti, J., Laippala, V., and Ginter, F. (2014). Syntactic n-gram collection from a large-scale corpus of internet finnish. In Grigonyt G. Kapočit-Dzikien J. Vaičenonien J. Utka, A., editor, *Frontiers in Artificial Intelligence and Applications (Volume 268): Human Language Technologies – The Baltic Perspective*, pages 184–191. IOS Press.

Loper, E. and Bird, S. (2002). NLTK: The natural language toolkit. In *Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics - Volume 1*, ETMTNLP '02, pages 63–70, Stroudsburg, PA, USA. Association for Computational Linguistics. `http://www.nltk.org/`.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In C.J.C. Burges, et al., editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc.

Przepiórkowski, A., Górski, R. L., Lewandowska-Tomaszczyk, B., and Łaźinski, M. (2008). Towards the National Corpus of Polish. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation, LREC 2008*, Marrakech. ELRA.

Řehůřek, R. and Sojka, P. (2010). Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta, May. ELRA. `http://is.muni.cz/publication/884893/en`.

Ziółko, B., Skurzok, D., and Michalska, M. (2010). Polish n-grams and their correction process. In *4th International Conference on Multimedia and Ubiquitous Engineering*. IEEE, August.

7. Language Resource References

Institute of Computer Science at the Polish Academy of Sciences. (2010). *National Corpus of Polish*. Ministry of Science and Higher Education, Research-development project of the Ministry of Science and Higher Education.