LanguageCrawl: a generic tool for building language models upon common Crawl

Szymon Roziewski¹ · Marek Kozłowski¹

Accepted: 15 June 2021 / Published online: 5 August 2021
© The Author(s) 2021

Abstract The exponential growth of the internet community has resulted in the production of a vast amount of unstructured data, including web pages, blogs and social media. Such a volume consisting of hundreds of billions of words is unlikely to be analyzed by humans. In this work we introduce the tool **LanguageCrawl**, which allows Natural Language Processing (NLP) researchers to easily build web-scale corpora using the Common Crawl Archive—an open repository of web crawl information, which contains petabytes of data. We present three use cases in the course of this work: filtering of Polish websites, the construction of n-gram corpora and the training of a continuous skipgram 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. This paper focuses particularly on high computing efficiency by applying highly concurrent multi-tasking. Our tool utilizes effective libraries and design. LanguageCrawl has been made publicly available to enrich the current set of NLP resources. We strongly believe that our work will facilitate further NLP research, especially in under-resourced languages, in which 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 · Language Models · N-gram

✉ Szymon Roziewski
sroziewski@opi.org.pl

Marek Kozłowski
mkozlowski@opi.org.pl

¹ National Information Processing Institute, Warsaw, Poland

1 Introduction

The internet is the largest and most diverse collection of textual information in human history, and, covers almost all known subjects and languages. This makes it an appealing resource for extraction of large-scale corpora for language modelling. However, until recently, it was highly unlikely that language researchers in academia would have had access to the necessary infrastructure to process such a large amount of information when building a language corpus. With recent improvements in computing power, storage availability and powerful, highly efficient scalable processing and computing frameworks, it has now become feasible to build a large scale corpus using publicly available web-archives and commodity hardware—that which can be purchased from retailers, such as low-cost desktop computers.

The **LanguageCrawl** tool supports NLP researchers to readily form large-scale corpora of a given language, filtered directly from the Common Crawl Archive. The user is then capable of building an n-gram collection and training neural-network-based distribution model. We strongly believe that this work will prove beneficial to the linguistic community. The corpus gathered for a given language, precomputed n-grams and distributed word representations are valuable for many purposes, such as boosting the accuracy of speech recognition, spell checking or machine translation systems (Buck, Heafield, & van Ooyen, 2014).

In this work our primary objective is to illustrate some of use cases of LanguageCrawl. Straining out Polish websites from the Common Crawl Archive and subsequently building n-gram corpora along with the training continuous skipgram language model are the main functionalities. As far as we know, such a large-scale collection has not yet been formed from the Polish internet. We consider our results to be highly useful for further NLP research in the Polish language, and to enrich knowledge on the wider subject of NLP, by means of models and data. In addition, our collection of Polish websites is currently the largest open text repository of Polish web-oriented language, counting more than 22 billion words. By the time this paper is published, the source code and language models used will be publicly available.

2 Related work

2.1 Data sets

Polish reference textual data-sets have been built during last fifteen years by several institutions, and the most popular of those are described below.

The National Corpus of Polish¹ (Przepiorkowski, 2012; Przepiórkowski, Górski, Lewandowska-Tomaszczyk, & Łażinski, 2008), (NCP) is a shared initiative of four institutions: the Institute of Computer Science at the Polish Academy of Sciences (the coordinator), the Institute of Polish Language at the Polish Academy of

¹ <http://nkjp.pl/>.

Sciences, the Polish Scientific Publisher, PWN, and the Department of Computational and Corpus Linguistics at the University of Łódź. The initiative was registered as a research and development project by the Polish Ministry of Science and Higher Education. The list of sources for the corpora contains classic literature, daily newspapers, specialist periodicals and journals, transcripts of conversations, and a variety of short-lived and internet texts. The NCP dataset contains approximately 1.5 billion words. It is distributed under a GNU GPL license in TEI-compliant XML format.

PELCRA² (Polish and English Language Corpora for Research and Applications) is also a research group at the Department of English Language at the University of Łódź, which was established in 1997. The main research interests and activities of the group include corpus and computational linguistics, natural language processing, information retrieval and information extraction. The PELCRA Reference Corpus of Polish was developed between 1996 and 2005 (between 2003 and 2005 as a research project of the Polish State Committee for Scientific Research) to address the demand for a large reference corpus of Polish language for research and applications. The corpus contains approximately 100 million words, including both written (90% of the corpus) and spoken, and mainly contemporary language (95%). The corpus comprises texts related to the European Union: news articles from the web page of the Community Research and Development Information Service, and press releases from the European Commission, the European Parliament and the European Southern Observatory.

The Polish Sejm Corpus (PSC) is a collection of stenographic transcripts from sessions of the Polish parliament, totaling approximately 200-million words (Ogrodniczuk, 2012). It contains automatically generated annotation on various levels encoded in TEI format. In large parts, the transcripts represent quasi-spoken speech, which makes it a highly interesting resource.

Wikipedia, being one of the largest freely available text sources, has proved useful for text mining purpose. The most recent dump after text extraction from the XML files and subsequent corpus pre-processing produced approximately 380-million tokens.³

In Banón et al. (2020), the authors report on methods for building parallel corpora by web crawling. They make use of CommonCrawl for exploiting web statistics of potential websites to run targeted crawling. To create the parallel corpora, the document and sentence alignment are performed. The data sets were evaluated in machine translation tasks with BLEU scoring. They released corpora containing 223 million sentence pairs from around 150k website domains, for 23 EU languages. Although, the formed data set is highly imbalanced, 73% of sentence pairs comprising of five languages: French, German, Spanish, Italian, and Portuguese. The Polish part of ParaCrawl is 3.7 GB in size, and contains approximately 45 and 666 million sentences and words, respectively. The size of our data set which we analyze in this work is 166 GB in size and is formed out of 22 billion words, and more than 1.3 billion sentences.

² <http://pelcra.pl/>.

³ <https://pl.wikipedia.org/wiki/Wikipedia:Statystyki>.

In another study on internet web corpora based on CommonCrawl (Suárez, Sagot, & Romary, 2019), the filtered, classified by language, shuffled at line level data set is developed. The pre-trained FastText model is used for language recognition. The resulting multilingual (166 languages) corpora are 6.3 TB in size, have 800 billion words. The Polish segment of the OSCAR corpora consists of 6.7 billion words and is 47 GB in size.

There is a strong need for a Polish textual data-set according to the Web Society. We decided subsequently to propose a freely-available tool for constructing such a data-set and corresponding language models using the Common Crawl archive. The Common Crawl archive is built by the non-profit Common Crawl organisation founded by Gil Elbaz in California. The purpose of Common Crawl is to enable wider access to web information by manufacturing and supporting an open web crawl repository. The Common Crawl Archive⁴ (Mühleisen & Bizer, 2012; Smith et al., 2013) is a publicly available index of the web which contains petabytes of scraped web site data. The data is made available both as raw HTML and as text only files. The Common Crawl corpus includes billions of pages of web crawl data collected over eight years. Common Crawl offers the largest, most comprehensive—open repository of web crawl data on the cloud. Our Polish subset of CommonCrawl after pre-processing contains more than 22 billion words, making it the largest among known open and readily accessible Polish text repositories.

2.2 Language models

Vectors are common both in AI and in cognitive science. The representation of documents and queries as vectors in a high dimensional space was introduced by Salton and McGill (1983). The Vector Space was used to discover the frequency of specific words as a clue for representing the information contained in documents. The frequencies of words and phrases now play a crucial role in language modelling.

Models that assign probabilities to sequences of words are known as Language Models (LM)—a probability distribution over sequences of words. Given such a sequence of length m , an LM assigns a probability to the whole sequence. The simplest model assigns probabilities to sequences of words, the n -gram. An n -gram is a sequence of n words: a unigram is a single word like machine; a bigram is a two-word sequence like machine learning, and a trigram is a three-word sequence like recursive neural network. The term, n -gram is used to mean either the word sequence itself or the predictive model that assigns it a probability. The unigram language model is commonly known as the bag-of-words model. Technically, an n -gram model is a collection of sequences of words with the corresponding frequencies, in which probabilities are counted dynamically using those frequencies. Having a method of estimating the relative likelihood of different phrases is useful in many natural language processing applications, especially ones that generate text as an output from speech recognition or generation, machine translation, or spelling correction (Manning, Raghavan, & Schütze, 2008).

⁴ <http://commoncrawl.org/>.

There are a handful of studies on using Common Crawl Data for n-gram generation, which corresponds to the concepts and entities explored in NLP. One such study is presented by Kanerva, Luotolahti, Laippala, and Ginter (2014), and offers an overview of possible applications of Common Crawl Data. The authors obtained both linear and syntactic n-gram collection from a Finnish language internet crawl, and made them publicly available. Some technical issues were also highlighted, specifically those of raw textual data processing and language detection. Another study (Buck et al., 2014) concerning n-gram counts and language models formed on the Common Crawl Archive is broader in the number of languages analyzed. 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 provided 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 the BLEU score by between 0.5 and 1.5 BLEU points.

In Wołk, Wołk, and Marasek (2017) the authors build pentagram language models of contemporary Polish language based on the Common Crawl corpus. They claim that their model is more efficient than Google WEB1T n-gram counts (Islam & Inkpen, 2009), in terms of perplexity and machine translation. Throughout their work, they focus on processing text-only files. It is stated that dealing with structured HTML files is insufficiently beneficial since it is non-trivial, and demands a high number of normalization steps with excessive computing power included. The language models were built on Common Crawl data and evaluated by a measure of perplexity on a few test sets. The perplexities obtained range between 480 and 1471.

In Ziółko, Skurzok, and Michalska (2010) a few Polish corpora were studied, statistics for n-grams were computed, and a tool for manual n-gram error correction was proposed. The authors pre-processed the textual data more carefully with respect to Polish diacritical marks. They also used a morphological analyzer for better word recognition. The n-gram model obtained is used for automatic speech recognition.

The bag-of-words (bows) or bag-of-n-grams document representation reports surprising accuracy for text classification and other application, owing to its simplicity and efficiency. However, the bows also have many disadvantages. The words order is lost, and thus different sentences can have exactly the same representation, in cases in which the same words are used in different orders. Even though the bag-of-n-grams method considers the word order in short context, it suffers from data sparsity and high dimensionality. Moreover, the bag-of-words and bag-of-n-grams methods offer little guidance on the semantics of the words or more formally the distances between the words. The performance is usually dominated by the size of the bag-of-words. Thus, there are situations in which simple scaling up of the basic techniques will not result in any significant progress, and more advanced techniques must be utilized.

Due to the rapid development of machine learning techniques in recent years, it has become possible to train more complex models on much larger data-sets, and

these models typically outperform the simple ones. The most successful concept is likely that which uses distributed representations of words.

Distributional Semantic Models (DSMs) have recently received increased attention, together with the rise of neural architectures for the scalable training of dense vector embeddings. One of the most prominent trends in NLP is the use of word embeddings (Levy & Goldberg, 2014; Levy, Goldberg, & Dagan, 2015), which are vectors whose relative similarities correlate with semantic similarity. Such vectors are used both as an end in itself for computing similarities between terms, and as a representational basis for downstream NLP tasks, such as text classification, document clustering, tagging parts of speech, named entity recognition, and sentiment analysis. The attention dedicated to neural network based distributional semantics also growing substantially. The main reason for this is the highly promising approach of employing neural network language models (NNLMs) trained on large corpora to learn distributional vectors for words (Mikolov, Sutskever, Chen, Corrado, & Dean, 2013c). Neural network based language models significantly outperform n-gram models (Mikolov, Chen, Corrado, & Dean, 2013a).

Mikolov et al. (2013a), Le and Mikolov (2014) have recently introduced the skipgram and continuous-bag-of-words models, which act as efficient methods for learning high-quality vector representations of words from large amounts of unstructured text data. The word representations computed using neural networks are interesting because the learned vectors explicitly encode many linguistic regularities and patterns. The best known tool in this field is currently Word2Vec,⁵ which allows fast training on huge amounts of raw linguistic data. Word2Vec takes a large corpus as its input, and builds a vector space, typically of several hundred dimensions, with each unique word in the corpus represented by corresponding vector in the space. In distributional semantics, words are usually represented as vectors in a multi-dimensional space. The semantic similarity between two words is then trivially calculated as the cosine similarity between their corresponding vectors.

With regards to the Polish language, there are a few studies that make use of word embeddings.

During the CLARIN project, which produces vector representations of Polish words (Kędzia, Czachor, Piasecki, & Kocoń, 2016). Wawer (2015) utilizes Word2Vec in order to propose a novel method of computing Polish word sentiment polarity using feature sets composed of vectors created from word embeddings. Previous work on Polish language sentiment dictionaries demonstrated the superiority of machine learning on vectors created from word contexts, particularly when compared to the semantic orientation of pointwise mutual information (SO-PMI) method. This paper demonstrates that this state-of-the-art method could be improved upon when extending the vectors by word embeddings, obtained from skipgram language models upon the Polish language version of Wikipedia.

In morphologically complex languages such as Polish, many high-level tasks in NLP rely on accurate morphosyntactic analyses of the input. In Ustaszewski (2016),

⁵ <https://code.google.com/p/word2vec/>.

the author investigates whether semi-supervised systems may alleviate the data sparsity problem exploiting neural network based distributional models. His approach uses word clusters obtained from large amounts of unlabeled text in an unsupervised manner in order to provide a supervised probabilistic tagger with morphologically informed features. Evaluations on a number of datasets for the Polish language including the National Corpus of Polish, PELCRA, the Polish Sejm Corpus, and Wikipedia were conducted.

Each of the papers discussed above utilize a Word2Vec model built upon standard well-known Polish resources, but not upon the CommonCrawl archive. However, such surveys have been performed in other countries. The authors of Ginter and Kanerva (2014) train a Word2Vec skipgram model on pentagrams both from the Finnish language corpus extracted from the Common Crawl dataset, and the Google-Books n-gram collection. They concern primarily, but not exclusively, word similarity and translation tasks. In the first task, semantic and syntactic queries are used to return word similarities. The second task involves testing the ability of Word2Vec representation in simple linear transformation from one language vector space to another. Training Word2Vec models on a collection of n-grams has the advantage of compactness: speedup for an English language n-gram corpus is increased by nearly 400 times, which means that it is possible to do the computations even on a single machine.

We have decided to focus on the Polish language in this work, and present the following:

- LanguageCrawl (our NLP toolkit),
- A Polish n-gram collection,
- a Word2Vec model trained on Polish Internet Crawl, which is based on the Common Crawl Archive.

We provide deeper statistical information about the Polish language, and present n-gram counts and their distribution, which is built on the basis of the huge Polish Internet Crawl, and is much greater in size than the corpora analysed in Ziółko et al. (2010). Additionally, this study improves on previous works by utilizing an 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 an increase in the number of worker nodes.

3 The proposed approach

3.1 Source data

The Common Crawl Archive (Crouse, Nagel, Elbaz, & Malamud, 2008), which we processed using LanguageCrawl, is an open repository of textual web crawl information containing petabytes of data. The information crawl has been gathered since 2008. It is accessible to anyone on the internet either via freely direct download or commercial Amazon S3 service, for which a deposit is demanded.

Fig. 1 Functionality diagram. Data Fetching as a core module begins all processing and is followed by optional Language Detection. Afterwards, the fetched data is stored in a database. This early stage continues in a loop. Both facultative, Sentence Extractor and Spelling Corrector are capable of textual data enhancement. At the end of the process, N-gram Building and Word2Vec Training are deployed. The crossed-circle mark denotes an *or* operation

The crawl data is stored in the Web Archive (WARC) format.⁶ The WARC format retains and processes data from the Common Crawl Archive dump which can be as large as hundreds of terabytes in size and contains billions of websites in a more effective and manageable way. The raw crawl data is wrapped around the WARC format, ensuring a straightforward mapping to the crawl action. The HTTP request and response are stored, along with metadata information. In the case of HTTP response, the HTTP header information is stored. This allows a high number of inquisitive insights to be gathered. The website content collected takes the form of an HTML document. Another available extension is WAT Response format,⁷ which is in the form of JavaScript Object Notation (JSON) file. In fact, WAT file content is directly mapped from the corresponding WARC file. The same process is followed for the third obtainable extension, WET Response format,⁸ with some difference in shortage of header information and plaintext web content instead of HTML. As most NLP tasks require only textual data, and we have access to limited resources, under the influence of Wołk et al. (2017), we decided to construct our tool around WET files containing plaintext website content with a minimal amount of metadata. Our use-cases are based on corpora extracted from the January to July 2015 Crawl Archive, which is approximately 866 TB in size, and contains approximately 1.12 trillion web pages. Since we consider only WET files, we fetched 68 TB of compressed textual data. Although the amount of data processed is enormous, the Polish language constitutes only a tiny fraction of it: we estimated its value to be approximately 0.3%.

⁶ <https://www.loc.gov/preservation/digital/formats/fdd/fdd000236.shtml>.

⁷ <https://commoncrawl.org/the-data/get-started/>.

⁸ <https://commoncrawl.org/the-data/get-started/>.

Processing data from the Common Crawl Archive is a colossal task, which is severely limited by the internet bandwidth connection, and a bottleneck for data fetching. In our case it took several for weeks to download enough data to construct a reasonable Polish website corpora. Our LanguageCrawl toolkit provides highly concurrent actor-based architecture for building a local Common Crawl Archive, along with n-gram collection for a specified language. Other tasks such as textual data processing, cleaning, and training Word2Vec on the corpora are accomplished within the same aforementioned, resource-efficient system. The core of data fetching and the language detection module is implemented using the successful Akka framework based on the Actor Model paradigm. The LanguageCrawl tool is launched on an efficient computer to perform these tasks.

3.2 Modules

LanguageCrawl consists of three core modules along with three submodules. The foundational components of the system are essential for many of LanguageCrawl's features, including data fetching, n-gram building, and Word2Vec training. Figure 1 shows a functionality diagram. The key and optional traits are shown in the colors blue and pink, respectively. Data Storage is separated to enforce the boundaries of the Data Fetching abstraction layer. The first stage of processing implemented in the Data Fetching module is continually repetitive until all URLs linking to Common Crawl Archive resources have been processed. The deduplication process is performed based on URLs hash comparison, in order to locate a duplicated web pages (Zeman et al., 2017). In the next step, we invoke Sentence Extractor for the cleansing of textual crawl data. The aim of this procedure is to remove all text phrases which are both functional parts of a website and do not carry any meaningful content. We may also use Spelling Corrector, which aims to fix character encoding problems within a single word. Subsequently, N-gram Builder is called, which forms the desired n-grams for specified ranks. We are then able to utilize Word2Vec Training, with the ability to implement a specific setup for parameters. Two training algorithms, CBOW (which is a continuous-bag-of-words model) and skipgram, may be deployed. The other factors that can be detailed are as follows: the dimensionality of the feature vectors, the maximum distance between predicted and current word in a sentence, the number of worker threads, and skipping words with lower total frequency.

3.3 Actor model

Each monthly Common Crawl Archive contains data resources distributed into many 140 MB gz-compressed files in order to facilitate data processing. Since textual information resides in disjoint files, it is straightforward to process the data in parallel. Processing web-scale data requires not only the passing of millions of messages concurrently, but also handling multiple failures, such as data store unavailability or network failures. That is why, we decided to use the Akka framework: a high-performing, resilient, actor-based runtime for managing

Fig. 2 Message passing diagram. FileMaster (M) sends message to its FileWorkers ($w_1, \dots, w_N$) wrapped in the router w_R . FileWorkers needs Bouncers ($b_1, \dots, b_M$) as helpers for language detection and writing data to Cassandra Storage. b_R is a router for Bouncer Actors. Db is Cassandra Storage

concurrency. We selected the Scala language for implementation of the Data Fetching stage.

The theoretical underpinnings of the Actor Model, a mathematical model of concurrent computation, are described in Hewitt, Bishop, & Steiger (1973). In Akka, actors are organized in a hierarchical structure known as the actor system, which is equipped with supervisor strategy—a straightforward mechanism for defining fault-tolerant service.

Akka Framework

The major benefit of using the Akka Framework⁹ is actor-based concurrency. The actors, like in a hierarchical organization, unaffectedly form a stratified structure. They exist to the outside as actor references—objects which can be passed around at ease and without limitation. One actor, which is responsible for a specific functionality within the program, might divide a larger task into smaller, more manageable pieces. For this aim it creates descendant actors and manages them. Each actor must know its supervisor by means of its reference. After the completion of a task, the current actor sends a message back to its executive. The supervisor receives a notice which indicates the state of the sent request, i.e. success or failure, and depending on that outcome, decides which routine to perform. The most significant aspect is that it is impossible to explore inside an actor and acquire an image of its state from the outside, except when the actor intentionally releases that information itself. There are several supervision strategies, most of which are related to the nature of the work and the character of the failure. When a problem occurs during processing, the subordinate can be restarted with a cleansing of its internal state, a child actor might be stopped permanently, or the failure may escalate leading to a supervisor fault. On the other hand, the secondary actor might be resumed with its stored internal state. It is essential to see an actor as a part of a superintendence hierarchy, which has major implications on subordinates when handling the failure. For example, resuming an actor will resume all of its descendant actors, and the opposite is true when terminating an actor. When we restart an actor, we can terminate and start all of its subordinates again, or preserve child states to themselves. The creation and termination of actors occurs in an asynchronous manner, without blocking the supervisor. The actor object has several

⁹ <http://akka.io>.

useful characteristics. Namely, it encapsulates state, owns behavior and communicates with other actors by exchanging messages. The actor object embodies some variables which reflects the possible states it might be in. This data makes an actor valuable, and one which must be protected from misuse by other actors. Actor behavior is an action to be taken in reaction to the message received. Actors communicate by passing messages which are generally immutable. Those notifications are placed into the recipient's mailbox.

In the Akka actor model, there is no guarantee that the same thread will execute an identical actor for another messages. Thus, it is strongly recommended to use only immutable messages in order to prevent messages corruption. The actor's objective is to process messages, which can be thought of tasks to do wrapped around data. The mailbox model assumes each actor has only a mailbox, and brings together the sender and receiver. The letters are enqueued in time-order of dispatching. It means that messages from other actors might not have an order at runtime, since some stochasticity occurs during the distribution of actors across threads. Passing a multiplicity of messages to the same receiver from the same actor will result in the same sequence. When an actor terminates by failing without restarting, stopping itself, or being stopped by its master, it will relieve all resources and pass all messages from its mailbox to the system's dead letter mailbox. The actor's mailbox is replaced within the actor reference by a system mailbox. The main actor within the whole system is Actor System, which forms a hierarchical set of actors sharing the same configuration. It first creates descendant actors and manages them, providing a certain configuration which is common for the child actors. As a matter of fact, Actor System starts and ends the application and is the root of the system.

LanguageCrawl In *LanguageCrawl*, Actor System creates File Master, an actor responsible for initiating File Workers and providing them with tasks to be accomplished. In fact, File Master is the only actor which manages and distributes tasks to its subordinates. It processes a file containing a list of links to WET files belonging to a specific monthly Common Crawl Archive dump. Each crawl contains 32000 URLs on average. Having fetched all of the links, File Master dispatches these URL paths to its individual File Workers. In an effort to avoid context-switching we decided to limit the number of File Workers to the number of virtual cores available in the cluster on which the program was run. Figure 2 shows a message passing diagram to explain in more detail all important processes which occur during runtime. In the MainAp Scala object we create Actor System, and define the configuration for the whole system and data-base connection. Actor System begins the processing flow by sending a message Start Downloading to File Master, specifying a file from which the URL links will be fetched and the number of File Workers to be initiated. File Master creates routers for both types of actors, File Worker and Bouncer. A router can be considered as a pool containing available actors of a certain type. Afterwards, File Master processes the URLs linking to WET files wrapped in a compressed archive from the Common Crawl Repository, creates its File Workers, and feeds each of them with one URL until all links are fetched. File Master activates its subordinates by continually sending them the Process File message with a URL string. File Master asynchronously awaits until it receives the

message, Processing Finished from File Worker, with additional information wrapped around an object. Subsequently, the master is capable of dispatching a new order to the freed worker. In the meantime, other workers process the URLs allocated to them by downloading data and extracting textual content from zipped archives. In order to speed up processing, an iterator pipeline is created. Several operations on streams occur, including unzipping and creating a buffer reader iterator. Textual content is extracted on top of the pipeline. During iteration over textual content, individual web documents are recognized by pattern matching. The opening fragment of metadata serves as an indicator for that purpose. Then, File Workers wrap a textual WET file crawl around an object, and send specimens of each particular web page for language recognition to the available Bouncers. The number of Bouncers was set to 36—slightly more than the number of workers. There is little benefit in increasing that value significantly, but decreasing the number of workers leads to a lowering of processing efficiency. When a Bouncer receives a Please Let Me In message with a document for language detection, it uses an external native library with a Scala wrapper for this purpose. It takes the first 100 words from the content, and passes them on to the language detector. If the selected language is within our interest, the whole document is transferred to the database. For better non-locking performance, writing to storage is asynchronous. This was achieved by using Future traits. As Future is completed, two cases are handled: success and failure. In the case of success, the number of written documents is sent to the File Worker; in the case of failure, the logger keeps track of information about the issue.

Language Detection Other than downloading of the data itself, language detection constitutes the largest amount of executing time (Kanerva et al., 2014) during the collection of crawl data. Thus, we selected the language detector with respect to its speed and accuracy, along with the ease of utilizing it from the Scala framework. The language detection module uses a highly efficient and accurate package, optimized for space and speed, which is known as Compact Language Detector 2 (CLD2) package.¹⁰ The tool is based on a Naive Bayes classifier, which is a probabilistic model. The first 100 words from any document are used as the input for CLD2. The package also has several embellishments for enhancement of the algorithm. Web-specific words containing no language information were truncated, such as *page*, *click*, *link*, *copyright*. Repetitive words that could affect the scoring, such as *jpg*, *foo.png*, *bar.gif*, were filtered. The charset of text sent for language recognition to CLD2 must be converted to UTF-8 character encoding. As an output, the language detector identifies the top three estimated languages and their probabilities. Such an indicator might be significantly beneficial, especially when the web page under consideration is multilingual. CLD2 implementation includes three alternative token algorithms to provide better prediction accuracy for certain languages which have peculiar properties. In the majority of cases, language recognition is achieved by quadgrams scoring, whereas for semanto-phonetic writing systems and syllabic alphabets, the analysis is conducted on the basis of single letters scoring. The term, quadram denotes a sequence of four adjacent letters.

¹⁰ Language detection is conducted with the <https://github.com/CLD2Owners/cld2>.

Lowercased Unicode text consisting of marks and letters, without digits and HTML tags, is scored afterwards. The training corpus was constructed manually from selected textual web samples of each language, which were then automatically enlarged with precaution by a further 100 million web pages. Both letter sequences extraction and scoring use table-driven methods, schemes which allow the system to search for information quicker, simpler and more efficiently. The algorithm is optimized by means of computation speed and space, specifically Random-Access Memory (RAM).

CLD2 is ten times more time efficient than other language detectors (Lui & Baldwin, 2012), is capable of recognizing 70 languages, and is wrapped in 1.8 M of x86 code and data structures. The primary lookup table offers coverage of 50 languages, relying on quadgrams, and consists of 256 K four-byte entities. The language detection performed on the average website containing 30 KB of HTML lasts only 1 millisecond on a current x86 CPU. Using the CLD2 package, one can identify 83 world languages. In one large-scale multilingual study (Zeman et al., 2017), the authors used this library for language identification.

After content written in a predefined language has been detected, Bouncers may perform optional spelling correction for words with character encoding problems¹¹ and subsequently enter the corrected content to the Cassandra Database.

Cassandra Storage We decided to take advantage of Apache Cassandra as the LanguageCrawl database, owing to its efficiency, scalability, and fault-tolerance. Apache Cassandra is an open-source project developed by the Apache Software foundation, which offers a host of interesting big-data-beneficial features. It is a distributed document database which supports a scaling trait, and is focused on lessening read/write latency (Rabl et al., 2012). Automatic data replication among computer nodes might be selected in the configurational setup. The adopted data model scheme is a hybrid between column-oriented and key-value pair models. Cassandra ensures a handy Application Programming Interface (API) known as Cassandra Query Language (CQL). A set of useful Cassandra connectors exists for many programming languages. Working with Akka for Scala, we utilized an open-source driver connector dedicated to Scala, developed by DataStax company.¹²

As our resources are limited, we built a two-node Cassandra Cluster for our study, containing two tables. The first stores a list of processed WET file links, and the second is the main table for crawl data. Having information about files already downloaded allows the system to be fault-tolerant.

4 Data set

The collected data-set contains nearly 55 million documents and more than 35 billion words, of which 12% and 5.3% have character encoding problems, respectively. The whole size of the textual corpora exceeds 407 GB. We performed

¹¹ The corrected words should be the same length as the original. A dictionary of correct words was built on a subset of literary texts taken from *National Corpus of Polish* (Przepiórkowski, et al. 2008), NCP.

¹² <https://github.com/datastax>.

Fig. 3 A pie chart showing the top-ten primary domain distribution across the documents. The first five entries represent more than 97% of the total. The majority of the documents are directly related to Polish primary domains. In particular, many entities with domains such as *com* and *org* have the subdomain *pl* e.g. *pl.wikipedia.org*, *pl.tripadvisor.com*

Primary Domains Distribution

Fig. 4 A pie chart representing the distribution of the top-ten domains across all documents. The first five entities constitute more than 80% of all domains

Domains Distribution

document processing in order to extract valuable content. Entries containing more than four sentences and without encoding problems were trained. Ultimately, 36.2% of the elements were omitted, resulting in a set containing 35 million documents, 22 billion words, and comprising 166 GB in total. We estimated the mean values for document length to be 741 words, or 4848 characters. A document in the data-set consists of 44 sentences on average. In one multilingual study (Zeman et al., 2017), the authors collected Polish Common Crawl corpora consisting of 5.2 billion words—the corpus used in this work is four times that size.

Figure 3 shows the top-ten primary domain distribution across the documents. The domains are well-concentrated and form three large clusters. The top-three domains contribute more than 90% of the whole set. Furthermore, one somewhat

Fig. 5 Curves representing language distributions over document fractions for Polish and six other languages are presented. In addition, a line for unknown (undetected) languages is shown. For the most part, the whole data-set consists of documents written in detectable Polish (the content of 55% of the documents comprises 97.5% Polish words. There is a considerable number of documents whose original language remains unknown (in 30% of documents, 15% of the words were unclassified)

surprising fact can be observed: among the top-ten primary domains, there is one which has German address. However, as 1.5 million Polish citizens currently live in Germany, this anomaly can be justified.

In Fig. 4 we present the distribution of the top-ten domains for documents. There are two major document sources within the data-set, namely *pl.tripadvisor.com* and *pl.wikipedia.org*. The first ten domains correspond to more than 99.9% all documents within the corpora.

We utilized of the CLD2 library to differentiate languages, as we mentioned in the section devoted to language detection. We provide a brief language distribution analysis into the collected data-set in order to ensure its coherency. The procedure is simple: we attempted to discover how many words to a particular language dictionary. We investigated all the vocabulary in the gathered corpora. Polish words comprise up to 76% of the dictionary, whereas 17% and 3% of entities are unclassified and English language, respectively. Five other languages Spanish, French, German, Russian, and Czech constitutes 2.7% of all words in the data-set. Entities not belonging to any language consist of a few types: fused words, misspelled words, or words which cannot be found in formal dictionary.

Figure 5 shows the language distributions across all documents. On the x axis we can see the document fractions. We selected several popular languages across the globe which utilize Latin alphabets, such as English, Spanish, French, and German. Two Slavic languages were added, due to their similarity to Polish language and their cultural proximity: Russian and Czech. Though the Russian language uses the Cyrillic alphabet, it is analyzed nevertheless, since it is a popular choice for Polish foreign language learners. It can be observed that approximately 55% and 49% of the documents contain 97.5% and 99% Polish words, respectively. On the other hand, 15% of the words in 30% of the entries were unclassified. The second most frequent language within the data-set is English, with 5% of the documents comprising nearly 40% English words. Entries with more than 50% of their words in

Table 1 Overview of samples of unique unigrams and their counts

Top 10		Middle segment	
Sample	Count	Sample	Count
Jest (is)	82,731,908	Pokorne (humble)	3201
Hotele (hotels)	37,044,318	Roślinności (vegetation)	3200
Więcej (more)	27,415,228	Łosi (moose in plural)	3200
Ma (to have in 3rd person)	22,112,466	Zabijano (to have been killed)	3200
Forum	22,070,464	Dojazdówka (diminutive access)	3200
Wszystkie (all)	21,041,834	Krokowych (related to step, adj)	3200
Tego (this)	19,571,471	Pseudonimami (form of pseudonyms)	3200
Może (can/maybe)	18,877,289	Boguchwałowice (Polish small town)	3200
Mam (I have)	16,988,514	Pozdówka (diminutive salute)	3200
Odpowiedzi (answers)	16,829,727	Pograj (imperative of play)	3200

other languages constitute less than 1% of the data-set. On the basis of this study, we can justify that CLD2 language prediction efficacy is well-founded.

5 Experiments

5.1 N-gram language model

This section offers deeper insights into the collection gathered, and presents the outcome of constructing n-gram corpora. Moreover, we report on several statistical characteristics, demonstrate examples of n-grams, and discuss the properties of the corpora in terms of weight and quantity presented with specific graph charts for better understanding.

Based on the Common Crawl Archive which had been scraped and processed, an n-gram model was constructed. After filtering out non-Polish web page content, data normalization was performed by lowercase conversion, sentence extraction using a tokenizer from the NLTK Python toolkit (Loper & Bird, 2002), removing all words containing non-alphanumeric characters, numbers, or encoding errors. We implemented statistical post-analysis to truncate fused and misprinted words. Also, words longer than twenty characters were erased before the creation of the language models. Polish diacritical marks and stop-words were preserved, since phrase search is enhanced by using them (Ziśko et al., 2010). After creating the n-gram model and data aggregation, all duplicated n-grams were removed. As a result, both n-gram set sizes and n-gram occurrences were reduced by approximately 54% and 60%, respectively (see both total # occurrences and collection size in Table 6 and 7). Meanwhile, eliminating fused and misprinted words diminished the data volume by another 31%. This indicated, that the raw crawl data was highly duplicated and contained useless content in many cases. Thus, we emphasize how important pre-

Table 2 Overview of samples of unique bigrams and their counts

Top 10		Middle segment	
Sample	Count	Sample	Count
Napisz, recenzje (write review)	4,155,761	Zdjęcia, gołe (naked pictures)	3401
Polityka, prywatności (privacy policy)	3,206,055	Frontera, hotele (frontera hotels)	3401
Adres, e-mail (email address)	3,098,369	Jedna, dawka (single dose)	3401
Proszę, czekać (please wait)	2,826,672	Polska, księgarnia (Polish bookstore)	3401
Więcej, informacji (more information)	2,736,751	Masz, ciekawy (you've interesting)	3401
Słowa, kluczowe (keywords)	2,620,938	Dwudziestu, pięciu (25—numeral)	3401
Restauracje, atrakcje (..., attractions)	2,553,696	Rodzenia, terapie (childbirth...)	3401
Plików, cookies (files cookies)	2,535,537	Krakow, powiat (Cracow district)	3401
Internet, explorer	2,475,434	Naszego, testu (of our test)	3401
Zobacz, więcej (see more)	2,382,278	Football, gdansk (football Gdańsk)	3401

processing is when working with the Common Crawl Archive. It is worth investigating how early deduplication, such as removing copyright notices, influences the resulting corpora. The N-gram Collection of the Polish Internet was established based on processing and analysis of an open repository Common Crawl Archive. It is formed from sets of unigrams, bigrams, trigrams, tetragrams and pentagrams. The starting point of this extensive analysis was to sort the n-gram collection in descending order with respect to entity occurrences, for each type of n-gram.

We provide n-gram examples of ten most frequently occurring entities for each n-gram type, and also a brief sample of unigrams, bigrams and trigrams from the collection.

Table 1 shows two lists of unique unigrams sorted by occurrence frequency. One list contains the top ten n-grams, whereas the second is related to mid-frequency elements. We present examples of Polish n-grams along with their English equivalents. Stopwords frequently open the entries, so we have removed them from the list to demonstrate some less obvious examples. The occurrences decline significantly when the index position increases. The tail of the collection is unprofitable, and is made of redundant words, mainly misprints and fused words, e.g. unigrams with frequency less than ten. Thus, they are not included in this table.

In Table 2 we present the ten most frequently chosen bigrams, not including stopwords. Most of the ten bigrams are formed from commonly used internet phrases, and their counts drop much slower than in the case of unigrams. The middle segment consists mostly of somewhat idiomatic expressions. The tail of the collection for rare bigrams is not shown.

Table 3 contains two lists analogous to the previous examples. The chosen entities are also presented without stopwords. The trigram count values are comparable. The most common of them are idiomatic phrases once again, and some of them are linked to corresponding bigrams from Table 2.

Table 3 Overview of samples of unique trigrams and their counts

Top 10		Middle segment	
Sample	Count	Sample	Count
Wikipedii, wolnej, encyklopedii	2,136,564	Canonical, ltd, dowiedz	3302
Internet, explorer, mozilla	2,021,722	Ceny, zdjęcia, wektory	3302
Explorer, mozilla, firefox	2,021,543	Kontakt, twoja, opinia	3302
Windows, internet, explorer	2,020,873	Zadawane, pytania, blog:	3302
Mozilla, firefox, google	2,020,341	Umowa, członkowska, licencja	3302
Witryna, tripadvisor, może	2,020,211	Strony, english, deutsch	3302
Tripadvisor, może, nie	2,020,131	z, komórki, język	3302
Nie, być, wyświetlana	2,020,082	Informacje, prawne, umowa	3302
Obsługiwane, są, następujące	2,019,952	Editorial, wideo, wszystkie	3302
są, następujące, przeglądarki	2,019,876	Język, strony, english	3302

Table 4 Overview of samples of unique tetragrams and their counts

Top 10	
Sample	Count
Internet, explorer, mozilla, firefox	2,080,495
Tripadvisor, może, nie, być	2,080,275
Witryna, tripadvisor, może, nie	2,080,143
są, następujące, przeglądarki, windows	2,079,846
Mozilla, firefox, google, chrome	2,079,428
z, wikipedii, wolnej, encyklopedii	2,079,390
Zauważyliśmy, ze, używasz, nieobsługiwanej	2,078,256
ze, używasz, nieobsługiwanej, przeglądarki	2,078,189
Może, nie, być, wyświetlana	2,077,754
Nie, być, wyświetlana, poprawnie	2,077,235

The top ten tetragrams and pentagrams are illustrated in Table 4 and 5, respectively. It is noticeable that the top tetragrams are somewhat associated semantically—a statement might be inferred from reading any of them. On the other hand, the most frequent pentagrams are generally composed of the top tetragrams and they are related to each other. The diversity in n-gram occurrences is hardly noticeable.

Table 6 yields an overview of the corpora by the means of the total occurrences of n-grams and sizes of corpora given in GB with respect to n-gram type. The number of all words within the crawl data extends beyond 22 billion entities and 166

Table 5 Overview of samples of unique pentagrams and their counts

Top 10	
Sample	Count
Tripadvisor, może, nie, być, wyświetlana	2,223,587
Może, nie, być, wyświetlana, poprawnie	2,223,498
Obsługiwane, są, następujące, przeglądarki, windows	2,223,438
są, następujące, przeglądarki, windows, internet	2,223,410
Następujące, przeglądarki, windows, internet, explorer	2,223,381
Windows, internet, explorer, mozilla, firefox	2,223,314
Przeglądarki, windows, internet, explorer, mozilla	2,223,227
Internet, explorer, mozilla, firefox, google	2,223,221
Explorer, mozilla, firefox, google, chrome	2,222,997
z, wikipedii, wolnej, encyklopedii, skocz	2,185,113

Table 6 Summary of numbers total occurrences for all type n-grams and sizes of corpora

N-gram type	Total # occurrences	Collection size
Unigram	22,335,478,164	0.26 TB
Bigram	19,969,363,630	1.82 TB
Trigram	18,375,050,955	3.38 TB
Four-gram	17,099,346,331	4.48 TB
Five-gram	16,071,746,162	5.38 TB
Total	93,850,985,242	15.32 TB

Table 7 Overview of numbers of unique n-grams and sizes of corpora, for each n-grams type

N-gram type	Total # occurrences	Collection size
Unigram	3,122,459	50 MB
Bigram	2,614,875	90 MB
Trigram	3,813,750	135 MB
Four-gram	4,211,349	182 MB
Five-gram	3,648,194	232 MB
Total	17,410,627	689 MB

GB of uncompressed textual data. The full volume of all corpora exceeds 15 TB of unaggregated data.

Table 7 summarizes the n-gram collection obtained after data aggregation. The Polish n-gram corpora, reaches nearly 17 million entries. It contains more than 3 million unique words, and is nearly 700 MB in size. The numbers of occurrences for higher-ranked n-grams are significantly larger than in the case of unigrams.

A survey of the length of the various n-grams may be an interesting prospect for language researchers. We offer additional insights on the topic, beyond what can be found in the typical dictionary, thanks to the larger size of the corpus we analyzed, the statistical measures we provided and the unique textual data source we gathered

Table 8 Statistics of unique n-grams' lengths

N-gram type	Mean	SE	Median	10th	90th
Unigram	8.89	2.44	9	5	13
Bigram	15.13	3.36	15	10	20
Trigram	23.20	5.46	23	15	31
Four-gram	31.45	6.84	31	22	40
Five-gram	38.71	8.12	39	28	50

Fig. 6 The n-gram curves show the counts of each n-gram type with respect to its length in characters, and visualizes the data from Table 6. The two maxima in the unigram line chart represent the stop-words counts and the most frequent size of Polish word, respectively. The usual length of Polish word is six characters. The curves of the n-gram occurrence functions for bigrams, trigrams, tetragrams and pentagrams form a right-skewed bell-curve, with long tails which widen as n-gram length increases. The chart curvatures lessen as the n-gram rank increases, and the maxima shift right by the value of 5

from websites. We believe that all of that may enrich understanding of the characteristics of Polish language, and shed new light on its analysis.

Table 8 presents several statistics pertaining to the n-gram length for each n-gram type, including the mean, standard error, median, and two percentiles: the 10th and the 90th. This data infers that the average Polish word is eight letters long, which would appear to be an overestimation due to the fact that a high number of fused words remain undetected in the n-gram corpora. Thus, it is reasonable to assume that all of our statistics are affected by this issue.

Fig. 7 Summary of the instances of each n-gram type for the 100 most frequent n-grams. The line charts demonstrate the variability in n-gram counts from the n-gram frequency table. The shapes of n-gram occurrence functions are steeper for unigrams, bigrams and trigrams, whereas for tetragrams and pentagrams the curvature is flatter, 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 remainder

Figure 6 shows n-gram occurrences for each n-gram type with respect to its length. The chart is based on much broader data than in paper (Roziewski, Stokowiec, & Sobkowicz, 2016), though the characteristics remain similar.

A few interesting phenomena concerning Polish n-gram corpora can be observed here: the number of occurrences have the same order of magnitude; for higher order n-grams, charts' shapes resemble bell curves, which broaden as the n-gram rank increases; the charts are more right-skewed and flatten with greater n-gram order; and the mean and median shift as the n-gram length increases, which is caused by the long right tail of the curves. The elongated tails for longer n-gram length may exist due to concatenation of undetected words. The chart depicting the distribution of unigrams is different from the others, as it demonstrates two local maxima. The first maximum may be the result of the existence of stop-words in the n-gram corpora, whereas the second, should be the most frequent word length in the Polish language based on Common Crawl Data, which is six characters. Stop-words occur frequently in the corpora: roughly 44% of the 274 frequently-occurring unigrams in our corpora are stop-words. We have estimated this ratio by comparing the numbers of the most frequent unique unigrams (only stop-words), and the Polish stop-words, extracted from the NLTK Python tool (Loper & Bird, 2002). The average length of those 44% of the unigrams is 3.32, which is reflected by the first lower maximum in Fig. 6. We can recognize that the maxima for higher-rank n-grams are shifted

Table 9 Perplexities for different n-grams

N-gram type	Median	Mean	SE
Bigram	771	2582	4187
Trigram	487	2145	4079
Four-gram	383	2362	4432
Five-gram	338	2519	4653

roughly by 6. We may infer from this fact that the average length of a Polish word is approximately that value, depending on the Polish n-gram corpora.

The charts from Fig. 7 present the occurrences of the top 100 n-grams for each n-gram rank analysed. The shapes of curves and the general characteristics are in accordance with results published in work (Roziewski et al., 2016) and remain intact.

It is noticeable that Zipf's Law manifests itself in the asymptotic decay of n-gram occurrences, especially in the case of the first three curves. The law states that the word occurrence in a corpora is inversely proportional to its index in the frequency list. The curves representing tetragrams and pentagrams are flatter. This may be caused by the fact that those n-grams are more highly correlated to each other among the n-grams within a rank.

5.2 Perplexity

Perplexity is a measure used for the evaluation of language models. It indicates how well a language model can predict a word sequence from the test set. It can be expressed by the formula:

$$PP(W) = \prod_{i=1}^N P(w_i | w_1 \dots w_{i-1})^{-N}, \quad (1)$$

where N , W and w_i are a sequence length, a word chain and w_i i th word, respectively.

We have not introduced any advanced smoothing methods as e.g. exponential smoothing methods. We cope with the problem of Out Of Vocabulary tokens using default minimal probability of appearance i.e. we performed replacement OOV tokens with $1/\mathcal{N}$ marginal value, where $\mathcal{N}$ is a number of all tokens in the training set.

We selected 10^5 documents at random to construct both sets, both for language model creation and its evaluation. The language model is built on the former set, and the measurement of perplexity was performed on the latter. The training set is three times bigger in size than the test set. We computed the perplexity measure without making use of smoothing.

In Table 9 the perplexities evaluated on the test set are shown. We calculated the mean, median, and standard error for the perplexities. The median value is far lower than the mean value, and the standard deviation is high. A useful example can be found in the prediction ranks. Each prediction's rank is based on the probability

estimated by the language model. In most cases, the prediction rank is 1, meaning that the model performed the best. However, in some cases when the model fails, it fails significantly, leading the perplexity mean and standard deviation values to rise substantially. Thus, the median value of the perplexity is effectively lower than its mean.

In Popel and Mareček (2010) the authors reported perplexities for bigrams and trigrams ranging from 368 (Catalan) to 3632 (Czech) and 325 (Catalan) to 3530 (Czech), respectively. Since Polish language is morphologically rich in a similar way to Czech, we can infer that our perplexities concur with Popel and Mareček (2010). In Wołk et al. (2017), the perplexities for Polish language models based on the Common Crawl corpus range between 480 and 1471. These are significantly lower values. However, the exact perplexity measure—neither the mean nor the median—is mentioned. In fact, in Wołk et al. (2017) deduplication was applied for removal of repeated sentences. Such an approach appears beneficial. Another perplexity study was performed in Buck et al. (2014) with mean values of 209 and 294 for the French and Spanish languages, respectively. In Ziolkowski and Skurzok (2011) the authors reported perplexity for Polish corpora ranging from 4258 (literature) to 16436 (Wikipedia), though the computational details were omitted.

5.3 Word2Vec

Word2Vec computes the distributional vector representation of words, and has been shown to help learning algorithms boost their scores in natural language processing tasks (Mikolov et al., 2013c). Continuous bag-of-words and skipgram 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, and parsing. The skipgram model is a method of learning word vector representations which is useful in predicting their context in the same sentence. The aim of the skipgram model is to maximize the function of average log-likelihood, given a sequence of training words $w_1, w_2, \dots, w_T$:

$$\frac{1}{T} \sum_{t=1}^T \sum_{j=-k}^k \log p(w_{t+j}|w_t), \quad (2)$$

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

$$p(w_i|w_j) = \frac{\exp(u_{w_i}^\top v_{w_j})}{\sum_{l=1}^V \exp(u_l^\top v_{w_j})} \quad (3)$$

where V is the size of vocabulary. The skipgram model featured with softmax is computationally expensive: computing $\log p(w_i|w_j)$ is proportional to vocabulary size, which can reach one billion. To boost the efficiency of the model, we use

Table 10 Examples of semantic similarities based on Word2Vec trained on the Polish internet corpora

Word	Most similar	Distance
Król (<i>king</i>)	Cesarz (<i>emperor</i>)	0.73
Tusk (<i>former PM</i>)	Donald (<i>his name</i>)	0.74
Kobieta (<i>woman</i>)	Dziewczyna (<i>girl</i>)	0.80
Mężczyzna (<i>man</i>)	Chłopak (<i>boy</i>)	0.85
Dziewczyna (<i>girl</i>)	Rozochocina (<i>horny</i>)	0.82
Apple	Tablety (<i>tablets</i>)	0.79
Dublin	Blessington	0.85
Sushi	Pizza	0.76

Table 11 Linguistic regularities in vector space

Expression	Nearest token	Distance
Samochód + rower (<i>car + bicycle</i>)	Motocykl (<i>Motocycle</i>)	0.71
Jezioro + las (<i>lake + forrest</i>)	Bagno (<i>Swamp</i>)	0.68
ptak – zwierzę + samolot (<i>bird – animal + airplane</i>)	Myśliwiec (<i>Fighter plane</i>)	0.65
sosna – roślina + zwierzę (<i>pine – plant + animal</i>)	Żubr (<i>Aurochs</i>)	0.60
król – mężczyzna + kobieta (<i>king – man + woman</i>)	Królowa (<i>Queen</i>)	0.58
dobry – zły (<i>good – bad</i>)	Najlepszy (<i>best</i>)	0.58

hierarchical softmax with lower computation demands bounded by $O(\log(V))$ as shown in Mikolov et al. (2013c).

In our service we used the Gensim Řehůřek and Sojka (2010) implementation of Word2Vec. Gensim implementation is fast enough to process the filtered corpus in less than a day. We set up the window size to 5 and the vector representations dimension to 300. In an effort to reduce memory usage, our training pipeline takes advantage of iterators. The model is trained in an on-line fashion, by fetching documents one after another from the database. The resulting model is approximately 5 GB in size, which makes it possible to train it even on machines with modest amounts of available RAM.

In Table 10 selected words with the entries closest to them in meaning are presented. The output is semantically coherent.

A few examples of linguistic computations based on vector space representation are shown in Table 11. Thanks to Word2Vec we have established a direct link between the mathematical representation of a word and its semantic meaning.

5.4 FastText

The FastText algorithm is a variant of Word2Vec approach. The origin Word2Vec model has the limitation of representing a distinct vector to each word, even if words are morphologically similar. The main difference is based on a new advance to the skipgram model, where each word is represented as a bag of character n-grams (Bojanowski, Grave, Joulin, & Mikolov, 2017). Thus, a word is expressed as a sum of these n-gram vector representations.

The problem to solve is independently predicting the presence (or absence) of context words for the word w_t . In Bojanowski et al. (2017), given the word at position t , the authors consider all context words as positive samples, and sample negatives randomly from the dictionary. The negative log-likelihood function based on the binary logistic loss is described by the formula:

$$\log \left(1 + e^{-s(w_t, w_c)} \right) + \sum_{n \in \mathcal{N}_{t,c}} \log \left(1 + e^{s(w_t, n)} \right), \quad (4)$$

where c is a context position, $\mathcal{N}_{t,c}$ is a set of negative samples from the vocabulary, n is a size of character n-grams, and s is a scoring function mapping pairs of (word, context) to scores in $\mathbb{R}$. The objective function (4) can be expressed as follows:

$$\sum_{t=1}^T \left[\sum_{c \in \mathcal{C}_t} \ell(s(w_t, w_c)) + \sum_{n \in \mathcal{N}_{t,c}} \ell(-s(w_t, n)) \right], \quad (5)$$

where ℓ is a logistic loss function $\ell : x \rightarrow \log(1 + e^{-x})$. The score s can be obtained as the scalar product between word and context vectors:

$$s(w_t, w_c) = u_{w_t}^T v_{w_c}. \quad (6)$$

The scoring function given by (6) is related to Word2Vec skipgram model with negative sampling (Mikolov et al., 2013c).

In the FastText model the function s is redefined as

$$s(w, c) = \sum_{g \in \mathcal{G}_w} z_g^T v_c, \quad (7)$$

where $\mathcal{G}_w \subset \{1, \dots, G\}$ is a set of n-grams appearing in w , G is a size of a dictionary of n-grams, and z_g is a vector representation for a given n-gram g . The word w is expressed as the sum of the vector representations of its n-grams. The scoring function (7) allows to share the representations across words, with the ability to learn vectors for rare words.

The FastText model can be trained in both ways, supervised and unsupervised. In Joulin, Grave, Bojanowski, and Mikolov (2017), the authors performed sentiment analysis and tag prediction. To speed up the running time, they used a hierarchical softmax to compute the probability distribution over the given classes (Goodman, 2001) with making use of the Huffman coding tree (Mikolov, Chen, Corrado, & Dean, 2013b). The computational time lessen from $O(kh)$ to $O(h \log_2(k))$, where k is

Table 12 Examples of semantic similarities based on FastText trained on the Polish internet corpora

Word	Most similar	Distance
król (<i>king</i>)	władca (<i>ruler</i>)	0.77
Tusk (<i>former PM</i>)	Tuska (<i>in a different case</i>)	0.74
kobieta (<i>woman</i>)	dziewczyna (<i>girl</i>)	0.75
mężczyzna (<i>man</i>)	facet (<i>guy</i>)	0.79
apple	myapple	0.69
Dublin	Dublinie (<i>in a different case</i>)	0.71
Sushi	Nigiri	0.72

Table 13 Word analogies in vector space generated by FastText

Expression	Nearest token	Distance
ptak—zwierzę + samolot (<i>bird—animal + airplane</i>)	Bombowiec (<i>Bomber plane</i>)	0.61
Moskwa — Rosja + Polska (<i>Moscow — Russia + Poland</i>)	Warszawa (<i>Warsaw</i>)	0.59
król—mężczyzna + kobieta (<i>king—man + woman</i>)	Królowa (<i>Queen</i>)	0.79

the number of classes and h the dimension of the text representation. In the sentiment analysis task they achieved comparable results to recurrent neural network LSTM-GRNN and outperformed convolutional neural network models, including CNN-GRNN (Tang, Qin, & Liu, 2015).

Since our data set is not labeled, we make use of an unsupervised training method. We used the FastText implementation from Facebook Open Source. We used the same model's parameters as for Word2Vec. The resulting FastText model is comparable in size to the Word2Vec model, though computation time was lower.

In Table 12, we present several nearest neighbours for given sample words. The results seem to be semantically coherent, though there is one thing to be addressed. For two given cases (Tusk, Dublin), the FastText model returned nearest neighbours which are types of Polish grammar cases. In particular, there is no such a possibility for the English language, where cases in grammar do not exist. One can apply stemming and lemmatization to reduce those drawbacks, especially for languages having complex grammar and morphology.

In Table 13, a few analogies are shown which are intact with those presented in Table 11. Although the analogies computed by the Word2Vec model seem to be more accurate for the Polish language which is morphologically complex.

6 Possible applications, conclusions and future work

In this paper we have presented LanguageCrawl—a linguistic tool which allows researchers to construct web-scale corpora from the Common Crawl Archive. We have also described and demonstrated its features and use cases. The tool is capable

of scraping the Common Crawl Archive with respect to a given language, and building both n-gram and distributional models using Word2Vec on the crawl data. It should be noted that the preprocessing of data when conducting analysis is essential. The n-gram model can be incorporated into speech recognition or a machine translation system to boost its performance, as demonstrated in Buck et al. (2014) and Wołk et al. (2017). Researchers may benefit from using a well-trained Word2Vec model on large-scale Polish corpora. The ability to use Elastic Search for fast-querying entities, return similar text to a certain query, and perform statistical analyses of textual corpora might prove outstandingly beneficial.

Statistics over a large Polish internet corpus provide a number of interesting insights. In this study, we have shown five Polish language n-gram type distributions, estimated the mean length of Polish words, and established their statistical characteristics.

The future direction of our work will involve equipping our toolkit with syntactic n-grams, and training Word2Vec on both linear and syntactic n-gram collections. We plan to evaluate the language models constructed in the machine translation task using the EUROPARL data set.

Acknowledgements We are immensely grateful to Dr. Irina Temnikova for her precious remarks and insights. We are grateful to Wojciech Stokowiec for his work as he is the author of the LREC'16 conference paper, which we are expanding in this study.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

- Banón, M., Chen, P., Haddow, B., Heafield, K., Hoang, H., Espla-Gomis, M., Forcada, M.L., Kamran, A., Kirefu, F., & Koehn, P., et al. (2020). Paracrawl: Web-scale acquisition of parallel corpora. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics* (pp. 4555–4567).
- Bojanowski, P., Grave, E., Joulin, A., & Mikolov, T. (2017). Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5, 135–146.
- Buck, C., Heafield, K., & 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.
- Crouse, S., Nagel, S., Elbaz, G., & Malamud, C. (2008). *Common Crawl Foundation*. <http://commoncrawl.org>
- Ginter, F., & Kanerva, J. (2014). Fast training of word2vec representations using n-gram corpora. In: E. Volodina, L. Borin, I. Pilán (eds.) *Linköping Electronic Conference Proceedings*. Uppsala University.

- Goodman, J. (2001). Classes for fast maximum entropy training. In *2001 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings* (Cat. No. 01CH37221) (Vol. 1, pp. 561–564). IEEE.
- Hewitt, C., Bishop, P., & Steiger, R. (1973). A universal modular actor formalism for artificial intelligence. In *Proceedings of the 3rd International Joint Conference on Artificial Intelligence, IJCAI'73*, pp. 235–245. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA. <http://dl.acm.org/citation.cfm?id=1624775.1624804>
- Islam, A., & Inkpen, D. (2009). Managing the google web 1t 5-gram data set. pp. 1–5.
- Joulin, A., Grave, E., Bojanowski, P., & Mikolov, T. (2017). Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics* (Vol. 2, Short Papers, pp. 427–431). Association for Computational Linguistics, Valencia, Spain . <https://www.aclweb.org/anthology/E17-2068>
- Kanerva, J., Luotolahti, J., Laippala, V., & Ginter, F. (2014). Syntactic n-gram collection from a large-scale corpus of internet finnish. In: G.G.K.D.J.V.J. Utka A. (Ed.), *Frontiers in Artificial Intelligence and Applications* (Vol. 268): Human Language Technologies—The Baltic Perspective, pp. 184–191. IOS Press.
- Kędzia, P., Czachor, G., Piasecki, M., & Kocoń, J. (2016). Vector representations of polish words (Word2Vec method) . <http://hdl.handle.net/11321/327>
- Le, Q., & Mikolov, T. (2014). Distributed representations of sentences and documents. In *Proceedings of International Conference on Machine Learning*.
- Levy, O., & Goldberg, Y. (2014). Neural word embedding as implicit matrix factorization. In *Advances in neural information processing systems* (pp. 2177–2185).
- Levy, O., Goldberg, Y., & Dagan, I. (2015). Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics*, 3, 211–225.
- Loper, E., & 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* (Vol. 1, ETMTNLP '02, pp. 63–70). Association for Computational Linguistics, Stroudsburg, PA, USA . <http://www.nltk.org/>
- Lui, M., & Baldwin, T. (2012). langid.py: An off-the-shelf language identification tool. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics* (pp. 25–30). Association for Computational Linguistics.
- Manning, C. D., Raghavan, P., & Schütze, H. (2008). *Introduction to information retrieval*. Cambridge: Cambridge University Press.
- Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013a). Efficient estimation of word representations in vector space. In *Proceedings of ICLR Workshop*.
- Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013b). Efficient estimation of word representations in vector space. arXiv preprint [arXiv:1301.3781](https://arxiv.org/abs/1301.3781)
- Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., & Dean, J. (2013c). Distributed representations of words and phrases and their compositionality. In: C. Burges, L. Bottou, M. Welling, Z. Ghahramani, K. Weinberger (eds.) *Advances in neural information processing systems* (Vol. 26, pp. 3111–3119). Curran Associates, Inc.
- Mühleisen, H., & Bizer, C. (2012). Web data commons-extracting structured data from two large web corpora. *LDOW*, 937, 133–145.
- Ogrodniczuk, M. (2012). The polish sejm corpus. In *Proceedings of the 8th International Conference on Language Resources and Evaluation* (pp. 2219–2223) (LREC 2012).
- Popel, M., & Mareček, D. (2010). *Perplexity of n-gram and dependency language models* (pp. 173–180).
- Przepiorkowski, A. (2012). Narodowy Korpus Języka Polskiego. Wydawnictwo Naukowe PWN
- Przepiorkowski, A., Górski, R.L., Lewandowska-Tomaszczyk, B., & Łażinski, M. (2008). Towards the National Corpus of Polish. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation*, LREC 2008. ELRA, Marrakech.
- Rabl, T., Sadoghi, M., Jacobsen, H., S., G.V., V., M.M., & Mankovskii, S. (2012). *Solving big data challenges for enterprise application performance management*.
- Řehůřek, R., & Sojka, P. (2010). Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks* (pp. 45–50). ELRA, Valletta, Malta . <http://is.muni.cz/publication/884893/en>
- Roziewski, S., Stokowiec, W., & Sobkowicz, A. (2016). *N-Gram Collection from a Large-Scale Corpus of Polish Internet* (pp. 23–34). Cham: Springer. https://doi.org/10.1007/978-3-319-30315-4_3

- Salton, G., & McGill, M. (1983). *Introduction to modern information retrieval*. New York: McGraw-Hill.
- Smith, J. R., Saint-Amand, H., Plamada, M., Koehn, P., Callison-Burch, C., & Lopez, A. (2013). Dirt cheap web-scale parallel text from the common crawl. *ACL*, 1, 1374–1383.
- Suárez, P.J.O., Sagot, B., & Romary, L. (2019). Asynchronous pipeline for processing huge corpora on medium to low resource infrastructures. In *7th Workshop on the Challenges in the Management of Large Corpora (CMLC-7)*. Leibniz-Institut für Deutsche Sprache.
- Tang, D., Qin, B., & Liu, T. (2015). Document modeling with gated recurrent neural network for sentiment classification. In *Proceedings of the 2015 conference on empirical methods in natural language processing* (pp. 1422–1432).
- Ustaszewski, M. (2016). *Data sparsity in highly inflected languages: the case of morphosyntactic tagging in polish*.
- Wawer, A. (2015). Sentiment dictionary refinement using word embeddings. In *International Symposium on Methodologies for Intelligent Systems* (pp. 186–193). Springer.
- Wołk, K., Wołk, A., & Marasek, K. (2017). *Big data language model of contemporary polish*. pp. 389–395.
- Zeman, D., Popel, M., Straka, M., Hajič, J., Nivre, J., Ginter, F., Luotolahti, J., Pyysalo, S., Petrov, S., Potthast, M., Tyers, F., Badmaeva, E., Gokirmak, M., Nedoluzhko, A., Cinková, S., Hajič jr., J., Hlaváčová, J., Kettnerová, V., Urešová, Z., Kanerva, J., Ojala, S., Missilä, A., Manning, C.D., Schuster, S., Reddy, S., Taji, D., Habash, N., Leung, H., de Marneffe, M.C., Sanguinetti, M., Simi, M., Kanayama, H., de Paiva, V., Droganova, K., Martínez Alonso, H., Çöltekin, Ç., Sulubacak, U., Uszkoreit, H., Mackentanz, V., Burchardt, A., Harris, K., Marheinecke, K., Rehm, G., Kayadelen, T., Attia, M., Elkahky, A., Yu, Z., Pitler, E., Lertpradit, S., Mandl, M., Kirchner, J., Alcalde, H.F., Strnadová, J., Banerjee, E., Manurung, R., Stella, A., Shimada, A., Kwak, S., Mendonça, G., Lando, T., Nitisaroj, R., & Li, J. (2017). CoNLL 2017 shared task: Multilingual parsing from raw text to universal dependencies. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies* (pp. 1–19). Association for Computational Linguistics, Vancouver, Canada.
- Ziolko, B., & Skurzok, D. (2011). *N-Grams Model for Polish*. <https://doi.org/10.5772/16568>.
- Ziółko, B., Skurzok, D., & Michalska, M. (2010). Polish n-grams and their correction process. In *4th International Conference on Multimedia and Ubiquitous Engineering*. IEEE, Cebu, Philippines.

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.