



F3

**Faculty of Electrical Engineering
Department of Computer Science**

Master's Thesis

Czech Foundational Large Language Model Corpus

Bc. Tommaso Gargiani
Open Informatics

May 2024

<https://github.com/tgargiani/Czech-LLM-Corpus>

Supervisor: Ing. Jan Šedivý, CSc.

I. Personal and study details

Student's name: **Gargiani Tommaso** Personal ID number: **483466**
Faculty / Institute: **Faculty of Electrical Engineering**
Department / Institute: **Department of Computer Science**
Study program: **Open Informatics**
Specialisation: **Data Science**

II. Master's thesis details

Master's thesis title in English:

Czech Foundational Large Language Model Corpus

Master's thesis title in Czech:

Český korpus pro velký jazykový model

Guidelines:

Investigate the latest research pertaining to the creation of a foundational large language model (LLM) corpus collection with a specific focus on Czech language. The primary goal is to design robust procedures for collecting and preprocessing a data corpus essential for training a foundational LLM. Explore diverse sources of internet data, including but not limited to platforms such as Common Crawl and Wikipedia. Focus on finding the most effective methods for cleaning and deduplicating raw text, trying to find a balance between efficiency and quality. Study the basic types of tokenizers and select an appropriate one for the Czech language. Implement scraping, downloading, and execute cleaning and deduplication procedures to construct a substantial corpus with a minimum of 100 billion Czech tokens. Evaluate the quality of the generated corpus through appropriate metrics.

Bibliography / sources:

- [1] Eldan, Ronen and Yuan-Fang Li. "TinyStories: How Small Can Language Models Be and Still Speak Coherent English?" ArXiv abs/2305.07759 (2023): n. pag.
- [2] Lee, Katherine, Daphne Ippolito, Andrew Nystrom, Chiyuan Zhang, Douglas Eck, Chris Callison-Burch and Nicholas Carlini. "Deduplicating Training Data Makes Language Models Better." Annual Meeting of the Association for Computational Linguistics (2021).
- [3] Penedo, Guilherme, Quentin Malartic, Daniel Hesslow, Ruxandra-Aimée Cojocaru, Alessandro Cappelli, Hamza Alobeidli, Baptiste Pannier, Ebtesam Almazrouei and Julien Launay. "The RefinedWeb Dataset for Falcon LLM: Outperforming Curated Corpora with Web Data, and Web Data Only." ArXiv abs/2306.01116 (2023): n. pag.

Name and workplace of master's thesis supervisor:

Ing. Jan Šedivý, CSc. Big Data and Cloud Computing CIIRC

Name and workplace of second master's thesis supervisor or consultant:

Date of master's thesis assignment: **23.01.2024** Deadline for master's thesis submission: **24.05.2024**

Assignment valid until: **21.09.2025**

Ing. Jan Šedivý, CSc.
Supervisor's signature

Head of department's signature

prof. Mgr. Petr Páta, Ph.D.
Dean's signature

III. Assignment receipt

The student acknowledges that the master's thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the master's thesis, the author must state the names of consultants and include a list of references.

Date of assignment receipt

Student's signature

Acknowledgement / Declaration

First of all, I would like to thank my supervisor Jan Šedivý for his support and guidance while working on this thesis and other projects. I am also grateful to my family for their constant support and belief in me.

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

Prague, May 24, 2024

.....

Abstrakt / Abstract

Tato práce představuje vývoj největšího českého korpusu pro trénování velkých jazykových modelů, jenž obsahuje 167 miliard tokenů. Jeho rozmanité zdroje dat zajišťují komplexní pokrytí českého jazyka. Ke zvýšení kvality našeho korpusu byly použity důkladné procedury čištění, filtrace a deduplikace. Natrénovali jsme také několik jazykových modelů na podmnožině korpusu, abychom demonstrovali jeho potenciál stát se pevným základem pro trénování českého velkého jazykového modelu.

Klíčová slova: český korpus, velký jazykový model, dataset, zpracování přirozeného jazyka

Překlad titulu: Český korpus pro velký jazykový model

This thesis presents the development of the largest Czech corpus for training large language models, comprising 167 billion tokens. Its diverse data sources ensure comprehensive coverage of the Czech language. Rigorous cleaning, filtering and deduplication procedures were employed to enhance the quality of our corpus. We also train several language models on a subset of the corpus to demonstrate its potential for becoming a solid foundation for training a Czech large language model.

Keywords: Czech corpus, large language model, dataset, natural language processing

Contents /

1 Introduction	1	4 Data Processing Pipeline	26
2 Language models	2	4.1 Data Cleaning and Filtering . . .	26
2.1 N-grams	2	4.1.1 Our Setup	27
2.2 Language Model Evaluation . . .	3	4.2 Deduplication	28
2.3 Tokenization	4	4.2.1 Exact Deduplication	28
2.4 Neural Language Models	5	4.2.2 Near Deduplication	29
2.4.1 Recurrent Neural Net- work (RNN)	6	4.2.3 Our Setup	29
2.4.2 Long Short-Term Memory (LSTM)	6	5 Corpus Statistics	31
2.4.3 Transformer	7	5.1 Detailed Statistics	31
3 Data Sources	9	5.2 Data Processing Pipeline Impact	32
3.1 Data Format and Schema	9	5.3 Data Size	33
3.1.1 JSON Lines	9	6 Model Training	35
3.1.2 Zstandard Compression . .	10	6.1 Training Time Estimates	35
3.1.3 Data Schema	10	6.2 Data Quantity, Quality and Model Performance	36
3.2 Common Crawl	10	6.3 Czech Models	38
3.2.1 Format	11	6.4 English-Czech Models	38
3.2.2 Spark and the cc- pyspark Library	12	7 Conclusion	41
3.2.3 Data Extract	13	References	42
3.3 HPLT	13	A Training Configuration	51
3.3.1 Data Extract	15	B Text Generation Examples	52
3.4 CulturaX	15	B.1 Czech Model 160M	52
3.4.1 mC4	16	B.2 English-Czech Model 160M . .	52
3.4.2 OSCAR	16		
3.4.3 Cleaning Procedures	17		
3.4.4 Deduplication Procedures .	18		
3.4.5 Data Extract	18		
3.5 MLP Books	18		
3.5.1 Data Extract	19		
3.6 iDnes	19		
3.6.1 Data Extract	19		
3.7 Wikipedia	20		
3.7.1 Data Extract	21		
3.8 Czech Sociological Review . . .	21		
3.8.1 Data Extract	21		
3.9 Patents	22		
3.9.1 Data Extract	22		
3.10 Theses	22		
3.10.1 Data Extract	23		
3.11 SYN v9	23		
3.11.1 Data Extract	24		
3.12 Plenary Speeches	24		
3.12.1 Data Extract	24		

Tables / Figures

3.1 HPLT text sizes across all languages	15	2.1 Hugging Face Open LLM Leaderboard	4
3.2 Distribution of document counts from mC4 and OSCAR in CulturaX’s initial dataset	16	2.2 Pseudocode implementation of BPE.....	5
3.3 Distribution of topics in SYN v9	23	2.3 Simple RNN architecture	6
4.1 Squeakily cleaner and filter thresholds.....	28	2.4 LSTM unit architecture.....	7
5.1 Detailed statistics of various data sources	31	2.5 Transformer architecture	8
5.2 Average number of words, sentences and paragraphs per record	32	3.1 Schema of a JSON object.....	10
5.3 Average number of words and sentences per paragraph	32	3.2 List of HPLT deliverables.....	14
5.4 Average number of words per sentence.....	33		
5.5 Impact of Data Processing Pipeline on word count	33		
5.6 Final data size of the Czech Foundational Large Language Model Corpus	34		
6.1 Training time estimates.....	36		
6.2 Test set perplexities of Czech models	39		
6.3 Test set perplexities of English-Czech models	40		
A.1 Training configuration of models	51		
B.2 Text generation configuration of models	52		



Chapter 1

Introduction

In recent years, the field of artificial intelligence (AI) has experienced an unprecedented surge, particularly in the domain of language models. These models, designed to understand and generate human language, have achieved remarkable advancements, revolutionizing various areas ranging from natural language processing to virtual assistants and machine translation.

However, despite the global diversity of languages and cultures, English language models have dominated the recent developments due to the vast availability of English corpora for pretraining. This neglect of other languages creates a significant gap in the accessibility of AI for non-English speaking communities, hindering the development of language-specific applications and solutions.

In response to this challenge, this thesis aims to address the gap by focusing on the development of a corpus specifically tailored for pretraining foundational large language models (LLMs) in the Czech language. Furthermore, we showcase the methodology employed in the creation of the Czech corpus, describing at length the data sources that compose it and the strategies adopted to ensure its quality and usability for pretraining LLMs.

The creation of such a corpus not only fills a void in the existing landscape of LLMs, allowing to fine-tune models pretrained on our corpus for other downstream tasks, but also facilitates the advancement of AI in Czechia.

Chapter 2

Language models

Language models are statistical models that assign probabilities to upcoming words, or sequences of words in general [1]. They are trained on textual data on the task of predicting the next word in a sequence based on the previous words – this is called *causal* language modeling. In this manner, language models learn the relationships and patterns within language, enabling them to compute the probability of a sequence and, consequently, to generate coherent text.

More formally, given a sequence of words w_1, w_2, \dots, w_n , a language model is a function that is able to compute either one of these probabilities:

$$P(w_1, w_2, \dots, w_n) \\ P(w_n | w_1, w_2, \dots, w_{n-1})$$

In the last few years, especially after the release of ChatGPT¹ in 2022, the term *large* language model (LLM) has become known to the general public. Although there is no clear definition of the term, a language model is often considered an LLM if it has been trained on large amounts of data with billions (or even trillions) of tokens and if it has a large number of parameters.

The definition of “large number of parameters” has shifted over the years. For instance, in OpenAI’s LLM family, the first model, GPT [2], had 117 million parameters. This number has increased to 1.5 billion parameters in GPT-2 [3], their second model, and to 175 billion in GPT-3 [4]. Even though there have been LLMs with even more parameters, such as Google’s PaLM [5] with 540 billion parameters, most open source LLMs released today have between 3 and 70 billion parameters.

2.1 N-grams

The joint probability of words w_1, w_2, \dots, w_n can be calculated using the chain rule of probability [6]:

$$P(w_1, w_2, \dots, w_n) = \prod_{i=1}^n P(w_i | w_1, \dots, w_{i-1})$$

Using the chain rule, we can see how the probability of the whole sequence is linked to the conditional probability of a word given its predecessors. However, calculating such probabilities is unfeasible, as there are too many possible sequences and a specific context might have never occurred before.

A solution to this problem is to approximate the context by using the Markov assumption, which says that the conditional probability of a word depends only on its predecessor:

$$P(w_n | w_1, \dots, w_{n-1}) \approx P(w_n | w_{n-1})$$

¹ <https://chatgpt.com>

A model that looks one word into the past, is called a bigram model. Consequently, we can generalize bigrams to models that look $n - 1$ words into the past – n-grams.

One way to estimate these probabilities is to use maximum likelihood estimation (MLE). In order to compute the MLE estimates, we need to count the number of occurrences of a word and its context, and normalize it by the number of occurrences of its context:

$$P(w_n|w_{n-1}) = \frac{C(w_{n-1}, w_n)}{C(w_{n-1})}$$

Despite their simplicity and computational efficiency, n-gram language models can work surprisingly well and are frequently used for tasks such as spelling correction or dataset filtering. However, they inherently struggle with capturing long-range dependencies and grasping semantics, especially when compared to neural network-based approaches.

2.2 Language Model Evaluation

In machine learning, it is common practice to split data into the training set and into the test set. After training the model with the training set, the test set is used to evaluate the model’s performance. Intuitively, the best language model is the one that assigns the highest probability to the test set.

Instead of evaluating language models using raw probabilities, it is more practical to compute the model’s perplexity, which is regarded as the standard metric for measuring language model performance. Perplexity is often described as a measure of “how much the model is perplexed by the test set”. A lower perplexity value indicates better performance.

Let w_1, w_2, \dots, w_n be a sequence of words (i.e. the test set). Perplexity is defined as the inverse probability of the test set, normalized by the number of words:

$$PPL(w_1, w_2, \dots, w_n) = \sqrt[n]{\frac{1}{P(w_1, w_2, \dots, w_n)}}$$

Normalizing by the number of words allows perplexity to be utilized for fair comparison between different texts of varying lengths. Perplexity can also be interpreted as the weighted average branching factor, indicating the average number of possible next words at each step of the prediction process. This interpretation naturally limits perplexity’s range to the interval $\langle 1, \infty \rangle$.

Nowadays, using only perplexity to assess the performance of LLMs with billions of parameters does not suffice, as it may not capture all aspects of their capabilities. In order to provide more comprehensive evaluations, LLM benchmarks are employed. They consist of standardized performance tests that assess various aspects of LLMs across different areas, such as commonsense [7] and mathematical [8] reasoning, language understanding [9], coding capabilities [10], etc. LLM benchmark results can be aggregated in leaderboards which aim to track the best models available at the moment. For example, the Hugging Face Open LLM Leaderboard² shown in Figure 2.1 scores open source LLMs on several benchmarks and ranks them by their average score.

As of May 2024, there are no publicly available LLM benchmarks for Czech models. Fortunately, there is a master’s thesis currently being written by our colleague Adam Jirkovský, which is focused on this topic.

² https://huggingface.co/spaces/HuggingFaceH4/open_llm_leaderboard

🤖 Open LLM Leaderboard

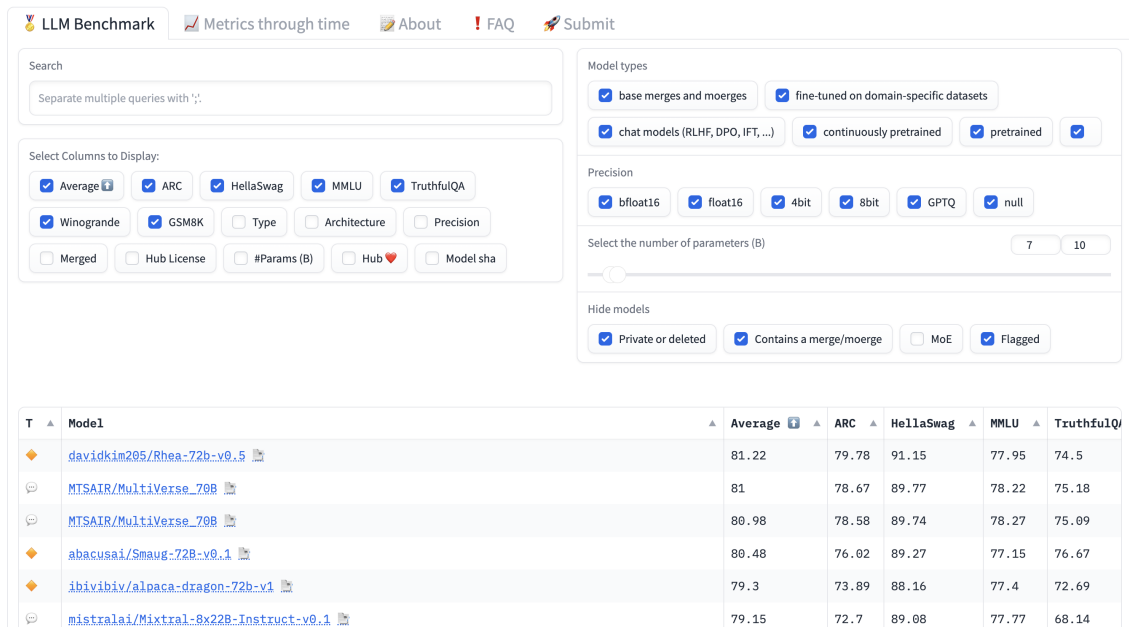


Figure 2.1. Hugging Face Open LLM Leaderboard.

2.3 Tokenization

In the previous sections, we have always described text as a sequence of *words*. However, it would be more adequate to define it as a sequence of *tokens*, where tokenization is the task of segmenting text into tokens. Tokens then may represent words, characters, or even subwords (parts of words).

Using subword tokens instead of words is useful when working with unknown words, which is an important problem in language processing [1]. Subword tokens can be arbitrary substrings of a word, but they are often a morpheme, i.e. the smallest unit of language that has its own meaning. For instance, the word *singers* contains three morphemes – the action *sing-*, the person doing the action *-er-* and the plural *-s*. This flexibility makes subword tokenizers especially valuable for languages with complex declension systems like Czech, where a single word can take many different forms, even though its root does not change. For instance, the first three cases of the word *bicycle* are: **kolo**, **kola** and **kolem**. Overall, subword tokenizers rely on the principle that frequently used words should not be split into smaller subwords, but rare words should be decomposed into meaningful subwords [11].

Byte-Pair Encoding (BPE) [12–13] is a popular tokenization algorithm used by many LLMs. The algorithm starts with a set of initial tokens, for instance all the ASCII characters, and a parameter k – the number of required merges. Then, the algorithm traverses the training corpus and counts the number of occurrences of all the neighboring tokens. After identifying the most frequent pair, the two tokens are merged across the whole corpus and a new entry is appended to the list of merges. This is repeated until k merges have been done. In the end, the new vocabulary consists of the original tokens and the k merged tokens.

See Figure 2.2 for the pseudocode implementation of BPE:

What we have described so far is character-level BPE, since the set of initial tokens were characters. However, this implicates that there also must be a token for unknown

```

function BYTE-PAIR ENCODING(strings  $C$ , number of merges  $k$ ) returns vocab  $V$ 
 $V \leftarrow$  all unique characters in  $C$            # initial set of tokens is characters
for  $i = 1$  to  $k$  do                           # merge tokens  $k$  times
     $t_L, t_R \leftarrow$  Most frequent pair of adjacent tokens in  $C$ 
     $t_{NEW} \leftarrow t_L + t_R$                  # make new token by concatenating
     $V \leftarrow V + t_{NEW}$                      # update the vocabulary
    Replace each occurrence of  $t_L, t_R$  in  $C$  with  $t_{NEW}$  # and update the corpus
return  $V$ 

```

Figure 2.2. Pseudocode implementation of BPE [1].

characters, because if we were to include all existing Unicode characters, the initial token set would be too large.

For this reason, byte-level BPE tokenizers are used. Instead of looking at words being written with characters, they look directly at bytes. This has two advantages. First, the base vocabulary is small, with only 256 tokens (the maximum number of values a byte can hold). Second, every possible character will be included without being converted to the unknown token.

2.4 Neural Language Models

Neural language models are trained in a self-supervised manner, where the model learns from a text corpus without the need for additional labeled data. In fact, since the language modeling task is to predict the next word in a sequence based on the previous words, the next word in the corpus serves as the implicit label for the sequence of words that precede it.

The language model is trained to minimize the error in predicting the next word, utilizing cross-entropy as the loss function. Cross-entropy measures the difference between the correct and predicted distribution. For language modeling, it corresponds to the negative log probability the model assigns to the correct word w_t , given the context of the previous n words:

$$L_{CE} = -\log P(w_t | w_{t-1}, \dots, w_{t-n+1})$$

Finally, we optimize cross-entropy using gradient descent-based algorithms.

The Transformer [14] is a deep learning model architecture that has driven the most recent advances in language modeling, with Transformer language models becoming synonymous with language models. Before its introduction, other neural network architectures such as recurrent neural networks (RNNs) [15] or long short-term memory (LSTM) [16] were used. The popularity of RNNs and LSTMs, as compared to simpler approaches such as n-grams or feedforward neural language models, is attributed to the fact that they addressed two important characteristics of language.

First, language is inherently sequential. Both RNNs and LSTMs take advantage of this, as they explicitly capture sequential dependencies between tokens, while feedforward language models process each token independently without considering its position relative to other tokens in the sequence.

Second, sentences may include interactions between words that have a long distance between them. For example, in the sentence “The **dog** I have just bought **is** an Akita.”,

the two words in bold are semantically connected despite being separated by intervening information. Since n -gram language models only have a limited context size of $n - 1$, it is impossible to use them for more sophisticated tasks that require longer context. Similarly, the context size of feedforward language models is limited, as it is bound to the size of their context window.

2.4.1 Recurrent Neural Network (RNN)

An RNN is a type of network characterized by containing at least one cyclic connection among its neurons, enabling the value of a neuron to depend on its own earlier outputs, either directly or indirectly, serving as an input. These recurrent connections allow the model to capture information about the sequence seen so far, which is stored in the model's hidden state. See Figure 2.3 for a simple RNN architecture with input x_t , hidden state h_t and output y_t at time step t .

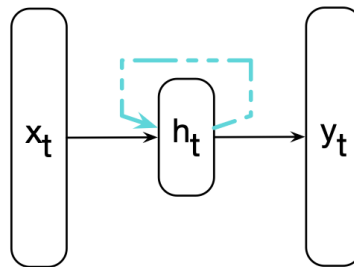


Figure 2.3. Simple RNN architecture [1].

RNN language models [17] process language sequentially, one word at a time – processing a sequence with t words requires t distinct time steps. They work by predicting the upcoming word from the current word and their previous hidden state. This is particularly convenient for language modeling, as the hidden state can represent context all the way back to the beginning of the sequence.

In practice, recurrent neural networks do not have an infinite context size, as they are constrained by the length of the sequences they are trained on. Furthermore, they suffer from the vanishing gradient problem during backpropagation, as the hidden layer from time step $t - 1$ contributes to the computation of loss at time step t . For long sequences, this leads to many multiplications and eventually drives the gradient to zero.

The information stored in their hidden state also tends to be more relevant for the recently processed part of the word sequence, struggling to propagate critical knowledge from its beginning. This is due to the fact that the hidden state needs to reflect current and future information at the same time.

2.4.2 Long Short-Term Memory (LSTM)

LSTMs are a variant of recurrent neural networks (they still retain the recurrent hidden layer) that address their problems with capturing complex context and vanishing gradient. They do so by selectively remembering information likely to be needed later and forgetting information that is no longer needed.

In an LSTM unit, this is achieved by keeping a memory cell, which retains information, and through the use of neural gates, i.e. specialized neurons that control the flow of information. In its computations, the unit operates with the input, the previous hidden state and the previous memory cell.

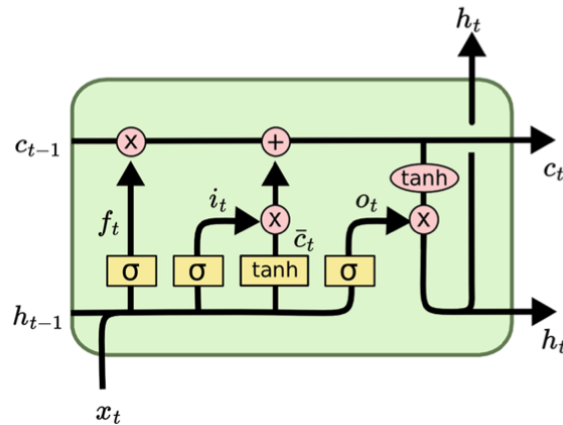


Figure 2.4. LSTM unit architecture [18].

See Figure 2.4 for the architecture of an LSTM unit with the input x , cell state c , hidden state h , forget gate f , add gate i and output gate o . As denoted by the vertical line leaving the unit on top, the current hidden state h_t at time step t serves as output.

There are three types of gates that share a common design – a feedforward layer with the sigmoid activation function, followed by a pointwise product with the layer being gated. The combination of the sigmoid (which pushes its outputs to 1 or 0) with the pointwise product acts similarly as a binary mask, controlling how much information gets through.

The forget gate is used to remove information from the cell state, while the add gate serves the opposite purpose. The third type, the output gate, decides what information from the memory cell is relevant to the current hidden state.

2.4.3 Transformer

Despite the improvements that LSTM brought to the original RNN architecture, both share a significant practical issue. They process sequences of data one step at a time, where the hidden state at each time step depends on the computations from the previous time step. This dependency creates a sequential bottleneck that makes parallel training difficult.

The key innovation of Transformer lies in its self-attention mechanism, which allows to weigh the importance of different words in a sentence when processing it. This attention mechanism enables Transformer to efficiently capture long-range dependencies between words in a sentence regardless of their positions in the sequence, allowing for parallel computation while remaining effective for tasks involving sequential data, such as language modeling.

As shown in Figure 2.5, Transformer consists of an encoder-decoder architecture, with multiple layers of self-attention mechanisms and feedforward neural networks. In the encoder, the input sequence is processed to create a contextualized representation for each word in the sequence. In the decoder, these representations are used to generate the output sequence, one word at a time, while attending to the relevant parts of the input sequence.

The authors of Transformer demonstrated the architecture’s potential on a translation, a sequence-to-sequence task, where the model receives the input sequence of words in the original language and outputs the translated sequence in a target language. However, the two halves of the encoder-decoder architecture can be separated to create models that use only one part, as they do not require the other half. Using

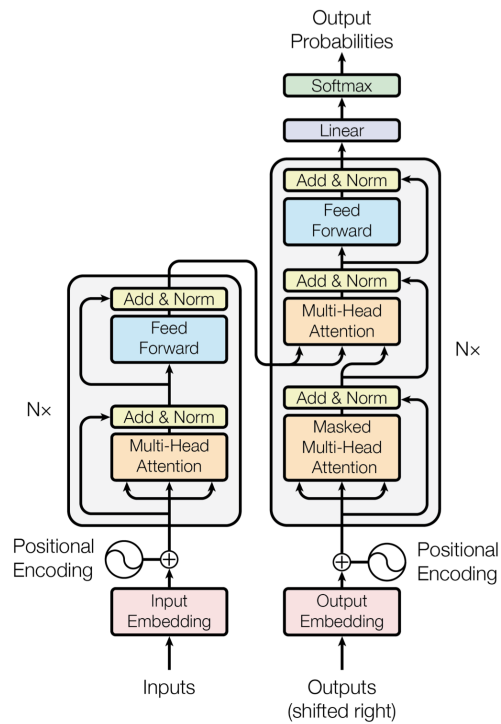


Figure 2.5. Transformer architecture with the encoder (left-hand side) and decoder (right-hand side) [14].

only the encoder is specific to models that require a good understanding of the input, e.g. the *masked* language model³ BERT [19]. On the other hand, decoder-only models are designed for generative tasks, such as text generation.

³ Masked language modeling (MLM) differs from causal language modeling in that it is trained to predict a masked (hidden) word within a sentence, rather than the next word. However, MLM is not the primary focus of this thesis.

Chapter 3

Data Sources

Our dataset is composed of several data sources. The sources range from generic web-pages crawled from the internet to more specialized content such as patents and parliamentary speeches. However, all these sources have one common trait – they are in Czech. Besides augmenting the overall dataset size, this approach ensures that the resulting dataset includes a diverse vocabulary and various writing styles, promoting the usability of the models trained with the dataset to multiple scenarios.

To focus on creating a robust foundational model, we have deliberately excluded instruction-based datasets at this stage. The initial goal is to develop a comprehensive understanding of natural language through a broad and diverse corpus. Including instruction-based data prematurely could skew the model towards task-specific patterns, limiting its versatility. This phased approach allows us to first establish strong general language capabilities, which can be fine-tuned with specialized datasets in subsequent training phases, ensuring balanced and adaptable model performance.

In this section, we will describe each individual data source in detail.

3.1 Data Format and Schema

All the data sources of the dataset are stored in the JSON Lines¹ format and compressed using Zstandard². This is a convenient choice, because both the JSON Lines format and Zstandard compression are commonly adopted in widely used NLP libraries, such as Hugging Face Datasets³ or Langchain⁴. For instance, in Hugging Face Datasets, the user can directly load training data from the compressed JSON Lines files by simply specifying the file path and the dataset schema.

3.1.1 JSON Lines

JSON Lines is a convenient format for storing structured data based on JSON, where every line stores a single JSON object. This feature makes `.jsonl` files streamable, allowing to process one record at a time.

JSON Lines has three requirements [20]:

1. **UTF-8 encoding:** using UTF-8 improves the readability of the stored data, as opposed to encoding Unicode strings with ASCII escape sequences (as standard JSON allows).
2. **Each line is a valid JSON record:** however, the `.jsonl` file as a whole is no longer valid JSON.
3. **Line separator is `\n`:** since JSON implicitly ignores white space outside of string literals, `\r\n` (used on Windows) is also supported.

¹ <https://jsonlines.org>

² <https://facebook.github.io/zstd/>

³ <https://huggingface.co/docs/datasets/index>

⁴ <https://www.langchain.com>

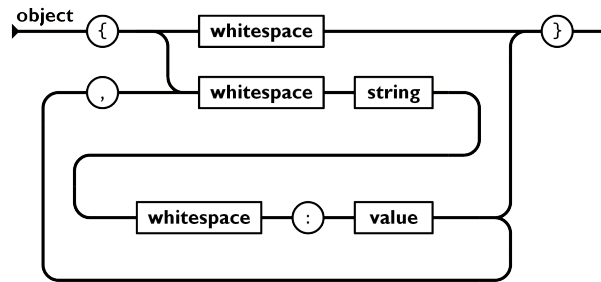


Figure 3.1. Schema of a JSON object [21].

JSON objects are unordered sets of name-value pairs [21]. As shown in Figure 3.1, an object is delimited by a curly bracket pair (`{...}`). A *value* is assigned to a *name* using a colon (`:`) and name-value pairs are separated by commas (`,`).

While names are required to be strings (a sequence of Unicode characters wrapped in `"..."`), values can be either strings, numbers, arrays (i.e. an ordered collection of values), `true/false/null` values or even different objects. Additionally, all these different structures can be nested.

3.1.2 Zstandard Compression

Zstandard (Zstd) is a lossless data compression algorithm [22]. Its library was released as open source by Meta (then Facebook) in 2016.

Zstd combines two types of lossless data compression methods – Finite State Entropy and Huffman Coding.

In its most recent version 1.5.6, it boasts a compression ratio 2.887, compression speed 510 MB/s and decompression speed 1580 MB/s [23]. These values have been calculated on an Intel Core i7-9700K CPU @ 4.9GHz system running Ubuntu 20.04.

3.1.3 Data Schema

There are some differences between our data sources as for what fields (names) are included in the individual JSON objects. For instance, the `url` field present in internet data sources is not relevant in objects containing books. Vice versa, the `author` field present in book data sources is not suitable for objects with internet content.

However, it is guaranteed that, across all documents, the `text` and `source` fields are present. The former includes, as the name suggests, the text of each document, usually split into paragraphs (delimited by the `\n` symbol). The latter denotes the code of the data source the document originates from, e.g. `commoncrawl`, `hpl1-v1.2` or `mlp-books`.

3.2 Common Crawl

Common Crawl⁵ is our dataset's main data source. It is a non-profit project which aims to maintain an open repository of web crawl data spanning from 2008 until today, accessible to anyone. Its corpus contains petabytes of data and can be accessed for free on the Amazon AWS S3 cloud platform.

A web crawler is a program that browses the internet automatically based on some predefined policies. Depending on implementation, a crawler may start with a list of URLs, store the content of each webpage and proceed in the same manner with all URLs found on the parsed webpages.

⁵ <https://commoncrawl.org>

Crawls are collected regularly, usually once a month, and contain the snapshot of a subset of the whole internet – including content older than the crawl’s timestamp. This means that, in case the same URL was selected for crawling and the webpage content has not changed, there may be an overlap between multiple crawls. The Common Crawl team calculated that approximately 65% of the unique URLs fetched in two subsequent crawls (e.g. June and July) are common to both.

■ 3.2.1 Format

The crawls are published in 3 different formats – WARC, WET and WAT.

WARC (Web ARChive) has become Common Crawl’s primary data format in summer 2013, replacing its predecessor – the ARC format. Proposed by the International Internet Preservation Consortium, it is used to store both primary (e.g. HTML) and secondary (e.g. metadata) content. WARC is an extension of the ARC format, which recorded only primary content. A WARC file concatenates multiple records (consisting of various text headers and a data block) into a single long file.

There are multiple types of records specified in the WARC specification. Common Crawl uses the `warc-type: response` type to store HTTP responses from the contacted websites. They also store information about the sent HTTP requests (`warc-type: request` type) and metadata about the crawl process (`warc-type: metadata` type)

Follows a shortened extract of the WARC data format, where the crawler fetched <https://en.wikipedia.org/wiki/Saturn> and received an HTML response:

```
WARC/1.0
content-type: application/http; msgtype=response
content-length: 583626
warc-ip-address: 208.80.154.224
warc-identified-payload-type: text/html
warc-date: 2023-09-29T08:25:05Z
warc-type: response
warc-target-uri: https://en.wikipedia.org/wiki/Saturn
warc-record-id: <urn:uuid:8007e174-e1f3-4778-90b9-70a4b776c64c>
...

HTTP/1.1 200 OK
date: Thu, 28 Sep 2023 16:42:36 GMT
server: mw-web.eqiad.main-644fddf9bf-xvvsz
x-content-type-options: nosniff
content-language: en
accept-ch:
vary: Accept-Encoding, Cookie
last-modified: Thu, 28 Sep 2023 16:41:57 GMT
content-type: text/html; charset=UTF-8
X-Crawler-content-encoding: gzip
accept-ranges: bytes
X-Crawler-content-length: 107286
Content-Length: 582140
...

<!DOCTYPE html>
<html lang="en" dir="ltr">
```

```
<head>
<meta charset="UTF-8">
<title>Saturn - Wikipedia</title>
...
```

As we can see, the WARC format contains extensive metadata, as well as a lot of HTML-related elements in its data block. However, for our goal of creating a Czech LLM corpus, these records are redundant and introduce unnecessary latency during data retrieval and processing.

Fortunately, the Common Crawl team also provides WET (WARC Encapsulated Text), a data format more suitable for our efforts. As shown in the following extract, WET only contains plain text extracted from the raw WARC data, as well as some selected metadata:

```
WARC/1.0
content-length: 80489
warc-record-id: <urn:uuid:df74c49c-f297-48e2-96de-533fa5068b73>
content-type: text/plain
warc-date: 2023-09-29T08:25:05Z
warc-type: conversion
warc-target-uri: https://en.wikipedia.org/wiki/Saturn
warc-refers-to: <urn:uuid:8007e174-e1f3-4778-90b9-70a4b776c64c>
warc-identified-content-language: eng
warc-block-digest: sha1:TPDGIBQ5NGM3333YFEHGT6K35P20ITTY

Saturn - Wikipedia
Jump to content
Main menu
...
```

3.2.2 Spark and the cc-pyspark Library

We use a modified version of the `cc-pyspark`⁶ library to download the data. As the name suggests, this library allows us to run a Spark job to download and process each record.

Apache Spark is a multi-language engine for executing data engineering, data science, and machine learning on single-node machines or clusters written in Scala [24]. Thanks to its distributed nature, it is ideal for processing large datasets such as Common Crawl. At the core of Spark is the Resilient Distributed Dataset (RDD). An RDD is a read-only collection of objects partitioned across a set of machines that can be rebuilt if a partition is lost [25]. RDD is an immutable distributed collection of objects that can be stored in memory across a cluster of machines. RDDs provide fault tolerance by keeping track of transformation operations to rebuild lost data partitions. They also support parallel operations like map, filter, reduce, and more, enabling distributed data processing at scale.

The whole Spark job defined in `cc-pyspark` can be summarized as follows:

1. **Downloading data from AWS S3:** the `boto3`⁷ library is used to fetch data from the S3 paths specified in the input file. This task is then distributed across the Spark cluster, ensuring efficient parallel retrieval.

⁶ <https://github.com/commoncrawl/cc-pyspark>

⁷ <https://boto3.amazonaws.com/v1/documentation/api/latest/index.html>

2. **Parsing records with FastWARC:** a high-performance C++ library used for parsing compressed WARC data streams.
3. **Processing data:** each record is filtered for Czech and a JSON string containing the record's text content and metadata is returned. The output file is a JSON Lines document created by merging all the stored JSON strings.

For downloading, we used an AWS EC2 *c7g.4xlarge* instance with 16 virtual CPUs. Thanks to using Spark, we took advantage of all the vCPUs and on average downloaded 585 MB of data per minute.

We have modified cc-pyspark to filter for Czech language documents by taking advantage of the *WARC-Identified-Content-Language* metadata tag. However, this tag is available in WET files only from the May/June 2020 crawl onwards. For this reason, we have only used crawls collected between May 2020 and June 2023, for a total of 23 different crawls. This tag has been provided by the Common Crawl team who ran the Compact Language Detector 2 (CLD2)⁸ on HTML pages to identify their language. Since CLD2 is able to identify up to 3 documents per document, we discard any records containing any other language than Czech.

The quality of crawled internet data varies significantly. Some webpages may contain high-quality texts, e.g. newspaper articles, while other may just be a nonsensical sequence of random symbols. In order to remove low quality or duplicate documents from our dataset, we process all Common Crawl data with our Data Processing Pipeline.

3.2.3 Data Extract

The JSON objects storing Common Crawl data have the following names (fields):

- **text:** the main content extracted from the webpage.
- **timestamp:** the timestamp indicating when the webpage was crawled.
- **url:** the URL of the webpage from which the data was extracted.
- **source:** the identifier for the data source, with the value set to `commoncrawl`.

Follows a shortened extract of the data:

```
{
  "text": "Zprávy - K mání je porevoluční sen mnohých Čechů, v originálním stavu už se dávno nevidí| AutoAdvert\nK mání je porevoluční sen mnohých Čechů, v originálním stavu už se dávno nevidí\nPokud jste \"automobilově vyrůstali\" v 90. letech, tohle auto jste nemohli ignorovat. Naprostá většina z nich je dávno zničená časem či tuningem, hezký originální kousek se ale pořád najde...",
  "timestamp": "2021-04-15T02:37:59.000",
  "url": "https://autoadvert.cz/magazine/2487115315-k-mani-je-porevolucni-sen-mnohych-cechu-v-originalnim-stavu-uz-se-davno-nevidi",
  "source": "commoncrawl"
}
```

3.3 HPLT

High Performance Language Technologies (HPLT) [26] is a joint project between several European universities and companies that aims to release large quantities of data in multiple languages to build powerful language models. The results of this project are to be published with open licenses.

As shown in their list of deliverables (see Figure 3.2), they plan to release not only training data for language models, but also cleaning software, trained models coupled

⁸ <https://github.com/CLD20wners/cld2>

Deliverables	0-12 Months	12-24 Months	24-36 Months	PDF
Initial release of monolingual and parallel data sets	✓ Lead by: UH - DONE - 12 months			↓
Final release of monolingual and parallel data sets	⚠ Lead by: UH - In progress - 35 months			—
Software for cleaning data sets	✓ Lead by: Prompsit - DONE - 18 months			↓
Clean filtered data sets augmented with metadata	⚠ Lead by: Prompsit - In progress - 36 months			—
First language models trained	✓ Lead by: UTU - DONE - 18 months			↓
Report on language model evaluation	⚠ Lead by: UTU - In progress - 35 months			—
Translation models for select language pairs	✓ Lead by: UEDIN - DONE - 18 months			↓
Report on evaluation of trained models	⚠ Lead by: CUNI - In progress - 35 months			—
Dashboard report	⚠ Lead by: Prompsit - In progress - 36 months			—
HPLT pipelines and tools	⚠ Lead by: VIO - In progress - 24 months			—
HPLT resource catalogue	⚠ Lead by: UH - In progress - 30 months			—

Figure 3.2. List of HPLT deliverables.

with some quality evaluations and other releases. These results are to be expected in their 36-month release plan.

So far, they have released three versions of their datasets. We use its third and latest version, where there are 75 monolingual and 18 bilingual datasets. Each dataset is available for download separately, based on the main language of its documents.

All three version are in the JSON Lines format and compressed using Zstandard. Besides the main text content, HPLT authors also provides other metadata, especially the document's main language (detected using CLD2), the URL of the document and even the languages identified at paragraph (line) level (using FastSpell [27]). Follows an extract from the original dataset's English release:

```
{
  "id": 1, "document_lang": "en",
  "scores": ["0.76", "0.76", "0.76"],
  "langs": ["en", "en", "en"],
  "text": "this is paragraph1\nthis is paragraph2\nthis is paragraph3",
  "url": "url1", "collection": "collection-1"
}
{"id": 2, "document_lang": "en",
 "scores": ["0.65", ...],
 "langs": ["en", ...],
 "text": "another paragraph\n...",
 ...
}
```

The first version contained only raw text data (as well as some metadata) with no data deduplication, or boilerplate removal and cleaning. The second version introduced near data deduplication done at document level using MinHash. We will talk about more data deduplication in Section 4.2.

Finally, some dataset cleaning has been performed on *HPLT Datasets v1.2*, the third release. A bug in their deduplication procedure has also been fixed. HPLT authors performed cleaning following 5 criteria [28]:

1. **URL is in the UT1 blacklist of adult sites**⁹: a collection of websites blacklists with more than 3.7 million records managed by Université Toulouse Capitole.
2. **An average of less than 5 words per line**
3. **Less than 200 characters in the document**

⁹ https://dsi.ut-capitole.fr/blacklists/index_en.php

4. Less than 5 lines in the document

5. Less than 20% of the lines in the document share the language identified at document level

Any document satisfying at least one of the criteria was removed. We also removed any non-Czech paragraphs, as identified by FastSpell.

The data in HPLT Datasets was acquired from two main sources: Internet Archive (crawls WIDE15, WIDE16 and WIDE17) and Common Crawl (crawl CC-MAIN-2022-40), which both contain internet webpage data. As shown in Table 3.1, Internet Archive, especially WIDE16, is the main source of HPLT’s data.

Crawl (collection)	# text files
CC40	127,853
IA WIDE15	495,512
IA WIDE16	977,792
IA WIDE17	798,811
Total	2,399,968

Table 3.1. HPLT text sizes across all languages [28].

After downloading the third release of the dataset, removing non-Czech paragraphs and converting the data to our schema, we proceed to process it with our Data Processing Pipeline.

3.3.1 Data Extract

The JSON objects storing HPLT data have the following names (fields):

- **text**: the main content extracted from the webpage.
- **url**: the URL of the webpage from which the data was extracted.
- **source**: the identifier for the data source, with the value set to `hplt-1.2`.

Follows a shortened extract of the data:

```
{
  "text": "Nekvalitní elektrokolo easy-bike od firmy Auto Kelly - Diskuse - Bike-forum.cz\nTento web používá k poskytování služeb, personalizaci reklam a analýze návštěvnosti soubory cookie. Používáním tohoto webu s tím souhlasíte. Více informací V pořádku\nNekvalitní elektrokolo easy-bike od firmy Auto Kelly\nNekvalitní elektrokolo easy-bike od firmy Auto Kelly\nPřed rokem a půl jsem koupil manželce elektrokolo easy-bike od prodejce Auto Kelly.\nKdokoliv toto čtete, prosím neudělejte nikdo stejnou chybu!\nJiž po roce baterie ztratila minimálně 1/4 své kapacity...",
  "url": "http://www.bike-forum.cz/forum/nekvalitni-elektrokolo-easy-bike-od-firmy-auto-kelly",
  "source": "hplt-1.2"
}
```

3.4 CulturaX

CulturaX [29] is a dataset tailored for large language model development, boasting 6.3 trillion tokens in 167 languages. Similarly to HPLT, the project aims to promote building open source language models in as many languages as possible, without being limited to English. This is due to the fact that despite there were several releases of

open source models, often only the model’s weights, not the training data, were made publicly accessible. Additionally, many of these models were specifically designed for English, limiting their potential global applicability in regions where English is not the dominant language, whereas more than 50% of CulturaX is dedicated to non-English languages.

It combines the mC4 [30] (version 3.1.0) and OSCAR [31–33] (distributions 20.19, 21.09, 22.01 and 23.01) datasets. Both mC4 and OSCAR are, yet again, internet content datasets, extracted from Common Crawl (CC). As we can observe in Table 3.2, mC4 makes up most of CulturaX’s initial data before cleaning and deduplication.

Dataset	% document counts
mC4	66
OSCAR 20.19	7
OSCAR 21.09	9
OSCAR 22.01	7
OSCAR 23.01	11

Table 3.2. Distribution of document counts from mC4 and OSCAR in CulturaX’s initial dataset [29].

As opposed to HPLT, the dataset’s original authors already perform extensive data cleaning and deduplication procedures similar to our Data Processing Pipeline. These procedures are crucial for the author’s goal of releasing an open source dataset which can be directly used to efficiently train language models of various languages. As discussed in Chapter 4, removing bad and duplicate content from the corpus improves the quality of language models, as well as reduces their training time, since we have fewer examples to go through during backpropagation.

■ 3.4.1 mC4

mC4 is constructed from 71 CC crawls by removing pages with less than 3 lines, less than 200 characters, or containing bad words. They also performed a more sophisticated version of sentence deduplication – all but one of any three-sentence span occurring more than once across all documents was removed. Subsequently, the corpus was exact-deduplicated at document level. The remaining pages are then grouped by language, as identified by CLD3 [34]. Documents with CLD3 confidence below 70% are removed.

■ 3.4.2 OSCAR

OSCAR is another corpus featuring multiple languages which were identified using fastText [35]. There are several key differences among OSCAR’s 4 distributions. For instance, versions before 22.01 are line-oriented, whereas the newer distributions become document-oriented. This means that the 21.09 version of an English Wikipedia article about the Czech anthem (with its lyrics) would be divided into the dataset’s both English and Czech subset, while the article is grouped with other English documents in the 22.01 version. Also, another advantage of the document-oriented approach is that the models trained on such data can have longer context, as the sentences are contiguous.

The dataset cleaning approach also differs among the various versions. They include, for instance, removal of documents with invalid UTF-8 characters, removal of short lines at either the head or tail of the document (only for recent releases), UT1 blacklist-based

filters, etc. They also chose to perform deduplication only on the early (i.e. sentence level) versions of the dataset. However, in contrast with CulturaX, dataset cleaning is not the main scope of the OSCAR project. Therefore, we omit the specifics of OSCAR’s filtering and proceed to the following section.

■ 3.4.3 Cleaning Procedures

As mentioned before, mC4 and OSCAR use different language identifiers. Since mC4’s CLD3 is substantially worse than OSCAR’s fastText [36], the authors chose to reclassify all mC4 data using fastText and discard any documents that did not correspond to the original language label.

They also discard any URLs present in the UT1 blacklist. This is due to the fact that mC4 did not include filtering based on the blacklist. Also, filtering once again OSCAR’s content allows the author’s to leverage the blacklist’s latest version.

Next, the dataset was cleaned based on a number of metrics, mostly inspired by the BigScience ROOTS corpus’s pipeline [37]:

- **Number of words**
- **Character repetition ratio**
- **Word repetition ratio**
- **Special character ratio**
- **Stop word ratio**
- **Flagged word ratio**
- **Language identification confidence**
- **Perplexity value**
- **Document length (number of characters)**
- **Number of lines**
- **Short line length ratio**
- **Short line ratio**

The perplexities are calculated using a 5-gram Kneser-Ney language model from the KenLM library [38] trained on Wikipedia, while the stop and flagged word lists from the BigScience ROOTS corpus are combined with those compiled by CulturaX’s authors. In order to classify a line as short, the threshold is set to 100 characters.

When setting the threshold of the metrics, the authors used a variant of the Interquartile Range (IQR) method [39]. After computing a distribution of all the metrics (for every language separately), they calculated the 10th and 90th percentiles of each distribution. The values at these percentiles become the thresholds for the aforementioned metrics – the lower percentile for metrics where high values are preferred (e.g. language identification confidence), and the higher percentile for metrics which favor low values (e.g. perplexity). All documents that fail to meet the set criteria are removed from the corpus.

After removing low-quality documents, the authors focus on improving the quality of the remaining data. As the dataset contains internet content, the previous steps may not be sufficient to completely remove all the noise from the data, such as leftover JavaScript (JS) code or website navigation menus. This noise is typically caused by crawling and extraction errors. Therefore, all short lines at the end of each document are removed, as they typically contain the website’s footer and other irrelevant content. All lines which contain at least two JS keywords (e.g. `<script`) are eliminated as well. However, if a document contains more than one such line, it is preserved, as it likely is a coding tutorial. The minimum number of JS keywords per line, so that it is considered

JavaScript, was set to two because of the resemblance some keywords bear to natural language (e.g. `var`). In this manner, the authors avoid unintentionally omitting helpful content.

■ 3.4.4 Deduplication Procedures

Deduplication is performed independently for each language in the dataset.

CulturaX is first deduplicated using the MinHashLSH near deduplication method, using a Spark implementation from the text-dedup¹⁰ GitHub repository. 5-grams and the Jaccard similarity threshold of 0.8 are used when configuring deduplication parameters. Near deduplication seeks to eliminate documents that have content similar to (though not necessarily identical with) other documents already existing in the corpus.

After near deduplication, the dataset is deduplicated according to website URLs. While crawling, multiple slightly altered versions of identical content may be retrieved using the same URL. These versions may differ sufficiently to evade near deduplication, yet not differ significantly enough to enhance the dataset's value.

Deduplication was the most resource intensive process when creating the dataset. It operated across 600 AWS EC2 instances, with each instance featuring 96 CPU cores, 192 GB of memory, and 1 TB of disk space. This disk space served as a substitute for memory when needed, such as during deduplication processes.

We will explore deduplication in greater detail in Section 4.2.

■ 3.4.5 Data Extract

The JSON objects storing CulturaX data have the following names (fields):

- `text`: the main content extracted from the webpage.
- `url`: the URL of the webpage from which the data was extracted.
- `timestamp`: the timestamp indicating when the webpage was crawled.
- `source`: the identifier for the data source, with the value set to `culturax`.

Follows a shortened extract of the data:

```
{"text": "Staročeské vdolky z plotny | Grafoman\n23.9.2013 Články, Pro volný čas, Tematické rubrikyLucie Švihovcová\nToto je perla staré české kuchyně. Recept mojí prababičky je prověřen několika generacemi nadšených strávníků.\nTento dezert potěší i dámy, které si hlídají štíhlou linii. Těsto je totiž zcela bez tuku či cukru...", "url": "http://www.m-grafoman.cz/clanky/staroceske-vdolky-z-plotny/", "timestamp": "2017/12/17 13:55:58", "source": "culturax"}
```

■ 3.5 MLP Books

This data source comprises books with expired copyright available from the Municipal Library in Prague. The books are usually available in 3 formats – EPUB, PDF and PRC. We have chosen to work with EPUB, a widely used file format for digital books, since it is easily machine readable by specialized libraries with little to no extraction errors. Hence, each book's content was extracted using EbookLib¹¹ and Trafilatura [40].

We have written a simple Python crawler that recursively explores all the subdirectories of `https://web2.mlp.cz/koweb/00`. Whenever the script encounters an `.epub`

¹⁰ <https://github.com/ChenghaoMou/text-dedup>

¹¹ <https://github.com/aerkalov/ebooklib>

file, it is downloaded as a temporary file and parsed using EbookLib. This enables us to extract metadata and individual chapters from the books. However, EbookLib presents book chapters in HTML format. To address this, we employ Trafilatura to extract plain text from HTML.

3.5.1 Data Extract

The JSON objects storing MLP Books data have the following names (fields):

- **title**: the title of the book.
- **text**: the whole book, with the book's chapters separated by `\n\n`.
- **author**: an array of the book's authors.
- **identifier**: the book's identifier.
- **source**: the identifier for the data source, with the value set to `mlp-books`.

Follows a shortened extract of the data:

```
{
  "title": "RUR",
  "text": "Karel \u010capek\nRUR\nROSSUM'S UNIVERSAL ROBOTS\nKOLEKTIVN\u00cd DRAMA O VSTUPN\u00cd KOMEDII A T\u0159ECH D\u011bjstv\u00cdch\nZn\u011bn\u00ed tohoto textu vych\u00e1z\u00ed z d\u00edla RUR tak, jak bylo vyd\u00e1no v \u010ceskoslovensk\u00e9m spisovatel\u00ed v roce 1994 (\u010cAPEK, Karel. Dramata : Loupe\u017e\u00edk : R.U.R. : V\u011b\u010d Makropulos : B\u00edl\u00e1 nemoc : Matka. 1. soubor. vyd. Praha : \u010ceskoslovensk\u00fd spisovatel, 1994. 472 s. Spisy, sv. 7.). Dal\u0161\u00ed d\u00edla Karla \u010c\u00e1pka naleznete online na www str\u00e1nk\u00e1ch M\u011bstsk\u00e9 knihovny v Praze: www.mlp.cz/karelcapek. Elektronick\u00e9 publikov\u00e1n\u00ed d\u00edla Karla \u010c\u00e1pka je spole\u010dn\u00fdm projektem M\u011bstsk\u00e9 knihovny v Praze, Spole\u010dnosti brat\u0159\u00ed \u010c\u00e1pk\u00fa, Pam\u00e1tn\u00edku Karla \u010c\u00e1pka a \u010cesk\u00e9ho n\u00e1rodn\u00edho korpusu.\n\u2022 OSOBY\nHARRY DOMIN, cent\u0159\u00e1ln\u00ed \u0159editel Rossumov\u00fdch Univerz\u00e1ln\u00edch Robot\u00fa\nIN\u017e. FABRY, gener\u00e1ln\u00ed technick\u00fd \u0159editel RUR\nDR. GALL, p\u0159ednosta fyziologick\u00e9ho a v\u00fdzkumn\u00e9ho odd\u011blen\u00ed RUR\nDR. HALLEMEIER, p\u0159ednosta \u0160stavu pro psychologii a v\u00fdchovu Robot\u00fa...",
  "author": ["Karel \u010capek"],
  "identifier": "urn:uuid:96185b52-4c47-43bb-b96e-8c9dfcb94f43",
  "source": "mlp-books"
}
```

3.6 iDnes

We have collected articles of the online newspaper iDnes from 1999 to 2023. They represent a reliable source of trustworthy high-quality data about current events from various categories, such as politics, sports, lifestyle etc.

To create the data, we created a Python scraper that fetches articles from their article archive. Using the library Beautiful Soup ⁴¹², we traverse the archive by date and find all the articles from that day. When found an article, we use Trafilatura to convert it from HTML to plain text.

3.6.1 Data Extract

The JSON objects storing iDnes data have the following names (fields):

- **title**: the title of the article.
- **text**: the content of the article.
- **date**: the article's publication date.
- **url**: the article's URL.
- **source**: the identifier for the data source, with the value set to `idnes`.

¹² <https://www.crummy.com/software/BeautifulSoup>

Follows a shortened extract of the data:

```
{
  "title": "Nejstrašidelnější knihovny světa, kde se budete bát nejen o Halloweenu - iDNES.cz",
  "text": "Willard Library (USA) Willard Library byla založena v roce 1885 a je nejstarší veřejnou knihovnou v Indianě. Tato historická knihovna sídlí ve viktoriánské gotické budově, kterou obývá duch Šedé dámy...",
  "date": "2023-11-01",
  "url": "https://www.idnes.cz/bydleni/na-navsteve/nejstrasidelnejsi-knihovny-sveta.A231031_142044_dum_osobnosti_rez",
  "source": "idnes"
}
```

3.7 Wikipedia

Wikipedia is a vast online encyclopedia that contains information on a wide range of topics, covering everything from history and science to popular culture and current events. It is collaboratively created and edited by volunteers from around the world, making it one of the largest and most comprehensive sources of information available on the internet.

Furthermore, Wikipedia stands out as an immensely popular data source incorporated into the training data of nearly every large language model available today. Wikipedia is known for its neutral tone and dedication to offering accurate, verifiable information, often supported by citations from reputable sources. These attributes make Wikipedia an exceptionally valuable resource for language models, offering a broad spectrum of topics to enhance their comprehension of various subjects.

All Wikipedia articles, grouped by language, are regularly collected as backup dumps in the XML format and compressed using bzip2. They are available for download at <https://dumps.wikimedia.org/backup-index.html>. We have used the November 2023 (20231101) dump.

All Wikipedia articles are written in Wikitext¹³, a wiki markup language for creating documents. Besides providing support for simple notation for text formatting (e.g. bold, italics, etc.), Wikitext also features more sophisticated editing constructs, such as links, tables, images, and various macros. Unfortunately, these constructs are also sometimes misused (e.g. unclosed brackets, missing attributes, etc.). Follows an example of the Wikitext markup language:

```
'''Florence''' {{efn|Obsolete [[Tuscan dialect|Tuscan]] form: {{lang|it| {{lang|it| Fiorenza}} {{IPA-it|fjo r n tsa|}}, from {{lang-la|Florentia}}.}} is the capital city of the [[Italy|Italian]] region of [[Tuscany]]. It is also the most populated city in Tuscany, with 360,930 inhabitants in 2023, and 984,991 in its [[Metropolitan City of Florence|metropolitan area]].<ref> {{Cite web |title=Bilancio demografico mensile |url=https://demo.istat.it/app/?a=2023&i=D7B |access-date=2023-04-25 |website=demo.istat.it}}</ref>
```

The data from the Czech version of Wikipedia was extracted and cleaned using WikiExtractor [41]. Not only WikiExtractor converts articles from Wikitext to plain text by using a set of regular expressions, it also uses a series of heuristics to circumvent the aforementioned bad formatting issues.

¹³ <https://www.mediawiki.org/wiki/Wikitext>

3.7.1 Data Extract

The JSON objects storing Wikipedia data have the following names (fields):

- **title**: the title of the article.
- **text**: the content of the article.
- **url**: the article's URL.
- **source**: the identifier for the data source, with the value set to `cswiki-20231101`.

Follows a shortened extract of the data:

```
{"title": "Astronomie", "text": "Astronomie, \u010desky t\u011b\u017e hv\u011bzda\u0159stv\u00ed, je v\u011bda, kter\u00e1 se zab\u00fdv\u00e1 jevy za hranicemi zemsk\u00e9 atmosf\u00e9ry. Zvl\u00e1\u0161t\u011b tedy v\u00fdzkumem vesm\u00edr\u00fdch t\u011bles, jejich soustav, r\u00fazn\u00fdch d\u011bj\u016f ve vesm\u00edru i vesm\u00edrem jako celkem.\nHistorie astronomie.\nAntika.\nAstronomie se podobn\u011b jako dal\u0161\u00ed v\u011bdy za\u010dala rozv\u00edjet ve starov\u011bku...", "url": "https://cs.wikipedia.org/wiki?curid=10", "source": "cswiki-20231101"}
```

3.8 Czech Sociological Review

Czech Sociological Review 1993–2016 [42] contains selected research articles and essays published in the Czech Sociological Review journal from 1993 to 2016.

It is a peer-reviewed sociological journal that has been publishing original contributions from Czech and international authors since 1965, covering a wide range of sociological topics. It features theoretical articles, analyses of social processes in post-communist and other societies with a focus on comparative analysis, overviews of developments in sociology and related fields, methodological studies, reviews, and other information about the sociological community. The journal is indexed in the Web of Science and other prestigious scientific databases. As an open-access journal, all content in the Czech Sociological Review is freely available to users and institutions. The journal is published six times a year, four times in Czech and twice in English, in both print and online formats. [43]

3.8.1 Data Extract

The JSON objects storing Czech Sociological Review data have the following names (fields):

- **title**: the title of the article.
- **text**: the content of the article.
- **author**: an array of the article's authors.
- **year**: the article's year of publication.
- **source**: the identifier for the data source, with the value set to `czech-socio-review`.

Follows a shortened extract of the data:

```
{"title": "Utv\u00e1\u0159en\u00ed nov\u00e9 soci\u00e1ln\u00ed regulace v \u010cesk\u00e9 republice", "text": "Panuje obecn\u00e1 shoda v tom, \u017ee p\u0159echod k tr\u017en\u00ed ekonomice mus\u00ed b\u00fdt siln\u011b soci\u00e1ln\u011b za\u0161t\u00edt\u011bn. Je t\u0159eba zabr\u00e1nit p\u0159\u00edli\u0161 hlubok\u00e9mu propadu t\u011b \u010d\u00e1sti populace, kter\u00e1 nen\u00ed schopna se adaptovat na nov\u00e9 ekonomick\u00e9 podm\u00ednky. V z\u00e1jmu prosazen\u00ed reformn\u00edch krok\u016f je t\u0159eba uchovat soci\u00e1ln\u00ed sm\u00edr. Soci\u00e1ln\u011b demokratick\u00e9 tradice p\u0159edv\u00e1le\u010dn\u00e9ho \u010ceskoslovenska a bl\u00edzkost \" soci\u00e1ln\u00ed Evropy\" br\u00e1n\u00ed n\u00e1vratu k\" divok\u00e9mu kapitalismu\" 19. stolet\u00ed...", "author": ["Ji\u0159\u00ed Ve\u010dern\u00edk"], "year": 1993, "source": "czech-socio-review"}
```

3.9 Patents

This is a collection of Czech patents downloaded from Czechia's Industrial Property Office¹⁴. They provide anonymized data in the XML format that follows the schema of the WIPO standard ST.96. We use the September 2023 database full export version of their data and disregard the subsequent incremental data.

Overall, patents are composed in highly technical language and are extremely domain-specific. This characteristic enables models trained on this data to acquire factual knowledge in areas often overlooked by other commonly used language modeling datasets. However, to construct a versatile language model, it's important to carefully balance the proportion of tokens originating from this source. This precaution is necessary to prevent the model from overly relying on excessively technical expressions during content generation.

The data was converted from XML to plain text by writing a conversion script that relies on the `xmldict`¹⁵ library. As the patents are split into thousands of XML files, the script operates by initially preprocessing the content of each XML file, removing certain formatting-related ST.96 tags. Subsequently, the XML document is converted to a Python dictionary using `xmldict`. Working with a dictionary simplifies the extraction of patent descriptions and other pertinent sections in accordance with their schema¹⁶.

3.9.1 Data Extract

The JSON objects storing Patents data have the following names (fields):

- **title**: the title of the patent.
- **abstract**: the abstract of the patent.
- **text**: the content of the patent's description.
- **source**: the identifier for the data source, with the value set to `patents`.

Follows a shortened extract of the data:

```
{
  "title": "Použití expandovaných činidel pro minimalizaci koroze a tvorby usazenin ve spalínových systémech",
  "abstract": "Způsob minimalizace koroze a vytváření usazenin na povrchu spalínového systému...",
  "text": "Vynález se týká způsobu minimalizujícího korozi, zejména zanášení bloků spalínových systémů a přidružených zásobních potrubí, kde je přítomno významné procento vlhkosti a /nebo kyselina sírová...",
  "source": "patents"
}
```

3.10 Theses

This data source includes all Czech language bachelor and diploma theses from the Faculty of Arts (Charles University) that are available at the university's digital repository.

We once again wrote a simple Python crawler that employs Beautiful Soup 4 to programmatically download the theses. Since they are in the PDF format, we have used the `pypdf`¹⁷ library to extract plain text from the documents. Although the extraction process worked fairly well, it is not perfect, as there are some imperfections

¹⁴ <https://isdv.upv.gov.cz/webapp/webapp.pubsrv.seznam?pid=11>

¹⁵ <https://github.com/martinblech/xmldict>

¹⁶ https://isdv.upv.gov.cz/doc/xsd/ST96/PatentPublication_V7_0.xsd

¹⁷ <https://pypdf.readthedocs.io/en/latest>

in the results, e.g. superfluous spaces within words, wrong characters etc. Due to the use of a PDF extractor, there are many paragraphs (separated by `\n`). We have opted not to remove them because there is no simple method to eliminate the superfluous `\n` without risking the unintentional removal of genuine paragraph content.

3.10.1 Data Extract

The JSON objects storing Theses data have the following names (fields):

- **title**: the (encoded) title of the thesis.
- **text**: the content of the thesis.
- **source**: the identifier for the data source, with the value set to `theses`.

Follows a shortened extract of the data:

```
{"title": "BPTX_2013_2_11210_0_382977_0_150571", "text": " \n Univerzita
Karlova v Praze \nFilozofická fakulta \nKatedra Psychologie \n \n \n
\nBakalářská práce \nHedvika Vertelmanová \nHumor jako protektivní
faktor profesní zátěže učitele...", "source": "theses"}
```

3.11 SYN v9

SYN v9 [44] is a corpus of written contemporary Czech, covering mostly the years 1990–2019. With 3,750,152 records, it contains a variety topics, including newspapers, fiction, non-fiction, etc. As we can observe in Table 3.3 (sorted by percentage in descending order), newspapers prevail noticeably:

Topic	%
Traditional newspapers	54.68
Leisure magazines	22.29
Prose	14.32
Shorter prose	2.44
Popular literature	2.34
Professional literature	1.17
Drama	0.91
Memoirs, (auto)biographies	0.86
Scientific literature	0.52
Poetry	0.36
Unclassified	0.09
Administrative texts	0.02

Table 3.3. Distribution of topics in SYN v9.

It is provided in a CoNLL-U-like vertical format:

```
<doc title="Velký Gatsby" author="Fitzgerald, Francis Scott" ...>
<block>
<s id="1">
" " " " Z:----- Z - -----
</g/>
Jsou Jsou být být VB-P---3P-AAI-- V - VDA3PP
```

```

to to ten ten PDNS1----- P 1 -----
staré staré starý starý AAFP1----1A---- A 1 -----
hodiny hodiny hodiny hodiny NNFP1-----A---- N 1 -----
<g/>
, , , , Z:----- Z - -----

```

When attempting to parse the data, we could not use any available CoNLL-U libraries, because of the minor differences between the two formats. Furthermore, we were not able to retrieve the documentation of this format the authors used. We were thus forced to write our own parser that, despite being equipped with only basic rules, proved effective in handling the data.

Using SYN v9 for model training has a significant disadvantage. The whole corpus is shuffled and divided into blocks of maximum 100 words, with ordering randomized within the given document. This is a significant limitation for training Transformer models. By segmenting the text into smaller blocks, the model is constrained to a context size of 100 words, which affects how it learns positional information and impacts the model's ability to capture long-range dependencies within the text.

3.11.1 Data Extract

The JSON objects storing SYN v9 data have the following names (fields):

- **text**: the content of the record.
- **title**: the title of the original work the record was extracted from.
- **author**: the original work's author.
- **txttype**: the topic of the record.
- **source**: the identifier for the data source, with the value set to `syn_v9`.

Follows an extract of the data:

```

{"text": "Tak jestli prý bych tam vyřídila, že ona dnes nemůže přijít?\n
Když Daisy mluvila, díval se na ni důstojník tak, jak si každá dívka
přeje, aby se na ni muž někdy díval, a já se dodnes pamatuju na tu
příhodu proto, poněvadž se mně to zdálo romantické.\n", "title": "Velký
Gatsby", "author": "Fitzgerald, Francis Scott", "txttype": "NOV: próza",
"source": "syn_v9"}

```

3.12 Plenary Speeches

This data source contains the speeches made at the Parliament of the Czech Republic from 1993 until 2023, as transcribed by their stenographers.

The original data was initially stored in CSV format and divided into two separate files. One file spanned the time frame from 1993 to 2016, while the other covered the years 2016 to 2023. Despite their overall high quality, the original datasets exhibited some flaws. The first file lacked punctuation, while the second file lacked capitalization. Therefore, we wrote a script that processed the speeches and addressed these grammatical inconsistencies using regular expressions.

3.12.1 Data Extract

The JSON objects storing Plenary Speeches data have the following names (fields):

- **text**: the content of the speech.
- **source**: the identifier for the data source, with the value set to `plenary-speeches`.

Follows a shortened extract of the data:

```
{"text": "Předseda PSP Milan Uhde Vážené paní poslankyně, vážení páni poslanci, vážené dámy a pánové, zahajují slavnostní schůzi Poslanecké sněmovny Parlamentu České republiky a všechny vás vítám. Zvláště srdečně vítám členy vlády České republiky, diplomatický sbor, představitele politického života, mezi nimi poslance bývalého Federálního shromáždění, a všechny čestné hosty. Scházíme se poprvé jako Poslanecká sněmovna Parlamentu České republiky, a to v první den její samostatnosti...", "source": "plenary-speeches"}
```

Chapter 4

Data Processing Pipeline

As we have mentioned before, internet crawled content contains a lot of low quality data. Some examples of low quality data include navigation menus, webpages that contain only lists of useless records, or lines that only contain long strings of random characters. Low-quality data hinders the language model’s learning process and its ability to generalize. Additionally, it is convenient to remove such content, as the extra tokens unnecessarily slow down the training procedure without bringing any tangible benefit. Removing potentially offensive or inappropriate content from the training data is also one of the most straightforward methods to control the model’s negative behavior.

Internet crawled content also contains a significant amount of duplicate data. Such data may be present due to various reasons – the URL could have been crawled multiple times, a website may repost content verbatim from another, changes in page formatting might have occurred without altering the content itself, etc. It is shown [45–46] that duplicate data negatively affects the language model’s performance, leading to memorization and requiring more training steps for the same or worse accuracy. Also, the duplicates may exist even between the train and test sets, potentially undermining the accuracy of the model’s evaluation.

Our pipeline operates through a two-step process. First, the data undergoes cleaning and filtering. Then, we perform text and URL deduplication.

4.1 Data Cleaning and Filtering

Although the terms data cleaning and filtering may seem interchangeable, we define cleaning as modifying the data (e.g. by removing some lines), whereas filtering as removing certain documents from the dataset.

We employ a modified version of Squeakily¹, an open source library for cleaning and filtering language datasets. It is a library inspired by the techniques used when creating The Pile [47], a vast English-only language modeling dataset, and BigScience’s ROOTS project, whose cleaning and filtering procedures also served as inspiration for those utilized in the CulturaX dataset.

To work with Squeakily, it is necessary to create a `Pipeline` object and define at least one `datasource` object, where we need to configure:

- **Dataset:** the dataset we are working with. Must be a `Dataset`² object from the Hugging Face Datasets [48] library.
- **Name:** the dataset’s name.
- **Columns:** since we are working with a `Dataset` object, a columnar data structure, we need to specify the columns to process.
- **Cleaners:** the functions to perform cleaning with.
- **Filters:** the functions to perform filtering with.

¹ <https://carperai.github.io/squeakily>

² https://huggingface.co/docs/datasets/v2.19.0/en/package_reference/main_classesdatasets.Dataset

While cleaners return the document’s cleaned version, filtering functions return a `true` or `false` value depending whether the document should be kept in the dataset.

After configuring the datasource(s), we need to create a `Pipeline` object that includes it. Then, we can run the pipeline. Since Squeakily works with `Dataset` objects, it leverages its multiprocessing capabilities, speeding up the data cleaning and filtering by parallelizing processes on the CPU.

By default, Squeakily executes the filtering functions first. However, we have deviated from this default behavior by specifying `cleaning_first=True` when running the pipeline. We made this decision to prevent documents from unexpectedly failing to meet the filtering criteria after undergoing the cleaning process. All the functions are processed in the sequential order they were defined in the datasource object.

We used the following cleaners:

1. **Remove empty lines**
2. **Normalize whitespace:** replaces various whitespace characters with the standard one.
3. **Fix UTF8 encoding:** replaces broken Unicode using the `ftfy` library [49].
4. **Remove short lines:** removes lines with less than 5 words from documents. It primarily aims to remove website navigation menus.
5. **Remove special character lines:** removes lines with a special character (i.e. punctuation and digits) ratio larger than 0.3. We have noticed that some pages contain undesirable random string sequences, likely due to formatting or extraction errors.

We implemented the functions from steps 4 and 5 after observing occasional occurrences of low quality lines in documents that are otherwise deemed acceptable.

After finishing the cleaning process, the following filters were run:

1. **Check number of words:** removes documents with less than 10 words. Too short documents are not likely to contain meaningful content.
2. **Check compression ratio:** removes documents that, when compressed with Zstandard, are smaller than a percentage of their original size (a set threshold). A low compression ratio value indicates that the document contains a lot of repeated content.
3. **Check flagged words ratio:** removes documents with an offensive word ratio larger than the set threshold. Aims to remove adult content.
4. **Check character repetition ratio:** removes documents that have a character repetition ratio larger than the set threshold. It is shown that repeated content has a negative effect on the performance of language models [50].

In order to set the thresholds present in steps 2, 3 and 4, we used the same variant of the IQR method that was used in CulturaX. We identify the 10th, 90th or 95th percentiles (depending on the metric) of the metrics’ data distribution, allowing to remove outlier documents that deviate from the norm.

■ 4.1.1 Our Setup

Since CulturaX was already cleaned by its authors using a pipeline similar to ours, we have cleaned only the other two internet content data sources we have collected – Common Crawl and HPLT.

The thresholds computed using the IQR method varied between the two data sources. See Table 4.1 for a comprehensive list of the utilized thresholds.

We deployed Squeakily on several AWS EC2 `c7g.4xlarge` instances (without clustering them), each one equipped with 16 vCPUs and 32 GB of memory, alongside AWS

Metric	Common Crawl	HPLT
Short lines	5	5
Special character lines ratio	0.3	0.3
Number of words	10	10
Compression ratio (0.05-q)	0.31	0.3
Flagged words ratio (0.9-q)	3e-4	2e-2
Character repetition ratio (0.95-q)	0.17	0.21

Table 4.1. Squeakily cleaner and filter thresholds.

EBS storage. During the cleaning process, only 10–20% of the available memory was utilized, while the computational load heavily leaned on the processor side, using all available vCPUs through parallel execution. On average, Squeakily managed to clean 593 MB of data per minute.

4.2 Deduplication

As previously mentioned, data deduplication plays a critical role in preparing language modeling datasets, as it enables the language model to focus on learning more generalized patterns and semantic structures from the data. We distinguish between two types of deduplication – *exact* and *near*.

Exact deduplication involves identifying and removing documents that are identical to others in the dataset, while near deduplication targets documents that are similar but not necessarily identical. Both approaches contribute to mitigating the over-representation of certain text segments in the training data, leading to more balanced and robust models

4.2.1 Exact Deduplication

Deduplicating large volumes of data (e.g. we have collected over 3 TB of Common Crawl data) demands substantial computational resources. Although open source deduplication libraries [45, 51] exist, they usually rely on distributing the process on a large cluster of instances, which we did not have at disposal.

To overcome this challenge, we have implemented a fast sequential C program that is able to run deduplication on a single machine, provided it possesses sufficient memory. It operates by calculating the hash of each document and maintaining records to determine whether the document has already been encountered. In theory, assuming no hash collisions occur, the program can distinguish between 8 billion unique documents per gigabyte of memory.

The program deduplicates all JSON Lines files within the specified directory and its subdirectories, passed as a command-line argument. Each file is read sequentially, with JSON objects extracted from each line utilizing the cJSON³ library. As output, it creates a new file that contains all the unique JSON objects.

The program first creates an array, where each bit is reserved to a unique document and represents if it has already been encountered. All bits are initially set to `false`.

Then, a hash (i.e. array index) is computed for each document. We compute the hash of a string using the following function:

³ <https://github.com/DaveGamble/cJSON>

```

unsigned long long compute_hash(const char *text,
                               unsigned long long max_unique_docs) {
    unsigned long long hash = 0;

    for (int i = 0; i < strlen(text); ++i) {
        hash = (hash * 127 + text[i]) % max_unique_docs;
    }

    return hash;
}

```

In the hash function above, we compute the document’s hash using Horner’s method, an algorithm for polynomial evaluation. Using the prime number 127 contributes to a uniform pseudorandom dispersion of hashes.

In case the document’s flag is set to `false`, the document is appended to a file containing only unique documents and its flag is set to `true`. Otherwise, the document is skipped.

■ 4.2.2 Near Deduplication

While various near deduplication methods exist, one of the currently most popular is MinHashLSH [52]. As the name suggests, it is technique based on MinHash [53] and Locality Sensitive Hashing (LSH).

Jaccard similarity is a frequently used indicator of similarity between two sets. In our scenario, we can imagine that a set contains our documents as elements.

Let U be a set, and A and B two subsets of U . Jaccard similarity is defined as:

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

Computing Jaccard similarity between two sets involves comparing all elements in the sets to determine the intersection and union of the sets. For large sets, this can be computationally expensive.

MinHash provides a way to approximate Jaccard similarity using a fixed-length signature, which is much faster to compute. This signature is computed by hashing the elements of the set using different hash functions and keeping track of the minimum hash value for each function. LSH then breaks each signature into bands, each band containing the same number of rows. If two documents share the same hashes in a band at a particular location (band index), they will be clustered into the same bucket and will be considered as duplicates. [54]

■ 4.2.3 Our Setup

Due to its high computational demands and the extensive volume of collected data, we did not perform near deduplication.

We ran exact deduplication only on the Common Crawl data source, using an AWS EC2 instance with 128 GB of memory, `r7g.4xlarge`. Our deduplication program ran on a single vCPU and used 110 GB of memory, having a theoretical upper limit of 880 billion possible unique documents. It processed 120 MB of data per minute. Overall, deduplicating ~ 1.7 TB of data took 10 days.

The other two internet content data sources, HPLT and CulturaX, were already deduplicated by their authors. Otherwise, we did not deem necessary the deduplication of the other manually collected data sources.

Even though Common Crawl, HPLT and CulturaX were all deduplicated, it was still possible that there were duplicates, because the data sources were deduplicated independently of each other. For this reason, we opted to run deduplication once more. Nevertheless, we have decided to run deduplication only on the URL fields of the documents, rather than their entire text content, following the approach used in CulturaX. URL deduplication helps to remove content that is likely only marginally altered, circumventing exact deduplication without contributing additional value to the dataset.

Chapter 5

Corpus Statistics

We have collected a substantial amount of data from a variety of sources, each distinct in several aspects. As discussed in Chapter 3 these sources differ in terms of language used, quality and the topics covered. Additionally, basic statistics such as the number of words and paragraphs vary among them. In this chapter, we will present the statistics of our corpus, including the impact of the Data Processing Pipeline on our internet-based data sources.

5.1 Detailed Statistics

We computed detailed statistics regarding the number of words, sentences, paragraphs and records regarding various data sources but the internet content based (Common Crawl, HPLT and CulturaX). We have excluded them because of their size and the computational cost associated with it. Furthermore, we do not report the paragraph count for the Theses data source, since PDF extraction leads to the occurrence of the paragraph delimiter `\n` even where it should not belong.

In order to compute the statistics, we used a library that provides various functions useful for working with text, NLTK¹. The function `nltk.word_tokenize` was employed to split text into a list of words, treating punctuation as individual words. This is accomplished through the use of regular expressions. For splitting text into sentences, we utilized `nltk.sent_tokenize`, which relies on the Punkt model [55]. To split text into paragraphs, the Python built-in function `string.split('\n')` was used.

In some cases, the number of paragraphs may exceed the number of sentences. This occurs because some sentences are not properly concluded with punctuation and instead end with the newline character `\n`. As a result, `nltk.sent_tokenize` fails to accurately detect the end of these sentences.

See Table 5.1 for the statistics.

Data source	# words	# sentences	# paragraphs	# records
MLP Books	147,330,265	7,707,452	4,316,116	2,665
iDnes	658,537,279	39,143,231	2,162,711	1,399,195
Wikipedia	178,487,122	10,132,490	4,243,174	528,716
Czech Sociological Review	3,105,462	120,842	522	522
Patents	549,699,573	19,962,867	10,339,532	233,552
Theses	590,910,380	29,299,118	/	14,126
SYN v9	5,670,228,847	341,843,113	362,603,997	74,788,660
Plenary Speeches	85,119,192	4,970,355	471,469	471,469

Table 5.1. Detailed statistics of various data sources.

¹ <https://www.nltk.org>

Then, for all data sources (including large – several GBs – extracts of Common Crawl, HPLT and CulturaX), we have also computed the:

- **Average number of words, sentences and paragraphs per record:** see Table 5.2.
- **Average number of words and sentences per paragraph:** see Table 5.3. Again, we omit the Theses data source for the aforementioned reasons.
- **Average number of words per sentence:** see Table 5.4.

Data source	\emptyset words	\emptyset sentences	\emptyset paragraphs
Common Crawl	470.8	19.6	30.9
HPLT	1,714.6	74.9	66.5
CulturaX	584.9	29.9	16.6
MLP Books	55,283.4	2,892.1	1,619.6
iDnes	470.7	28	1.5
Wikipedia	337.6	19.2	8
Czech Sociological Review	5,949.2	231.5	1
Patents	2,353.6	85.5	44.3
Theses	41,831.4	2,074.1	/
SYN v9	75.8	4.6	4.8
Plenary Speeches	180.5	10.5	1

Table 5.2. Average number of words, sentences and paragraphs per record. All numbers reported are rounded to one decimal place.

Data source	\emptyset words	\emptyset sentences
Common Crawl	15.2	0.6
HPLT	25.8	1.1
CulturaX	35.2	1.8
MLP Books	34.1	1.8
iDnes	304.5	18.1
Wikipedia	42.1	2.4
Czech Sociological Review	5,949.2	231.5
Patents	53.2	1.9
SYN v9	15.6	0.9
Plenary Speeches	180.5	10.5

Table 5.3. Average number of words and sentences per paragraph. All numbers reported are rounded to one decimal place.

5.2 Data Processing Pipeline Impact

In this section, we discuss the impact on size that the Data Processing Pipeline has on our data sources.

As mentioned before, we have run our Data Processing Pipeline only on internet content sources – Common Crawl, HPLT and CulturaX. Because of their size, it is impractical to calculate their size precisely. Instead, we used the `wc` (word count) Unix command to estimate that 1 GB \approx 135 million words.

Data source	\varnothing words
Common Crawl	24
HPLT	22.9
CulturaX	19.6
MLP Books	19.1
iDnes	16.8
Wikipedia	17.6
Czech Sociological Review	25.7
Patents	27.5
Theses	20.2
SYN v9	16.6
Plenary Speeches	17.1

Table 5.4. Average number of words per sentence. All numbers reported are rounded to one decimal place.

The results are summarized in Table 5.5. Its rows follow the steps of our pipeline, i.e. the reported size corresponds to the size after the data has undergone the respective step.

The rows in the last step, URL deduplication, are merged. This is due to the fact that it was performed on all data sources simultaneously, meaning that the result would be biased based on which data source the deduplication algorithm processed first. Therefore, we report the joint results for all three data sources.

Pipeline step	Common Crawl	HPLT	CulturaX
Original	544.05 (100%)	19.11 (100%)	28.89 (100%)
Cleaning and filtering	234.23 (43%)	12.52 (66%)	/
Deduplication	180.53 (33%)	/	/
URL deduplication	91.95 (16%)		

Table 5.5. Impact of Data Processing Pipeline on word count. All numbers reported are in billions of words and rounded to two decimal places.

5.3 Data Size

We report the final size of our corpus in Table 5.6 in both number of words and tokens.

When reporting the word count of the three internet content data sources, we use the estimate described in the previous section. Regarding the other data sources, we use the exact word count, as detailed in Section 5.1.

The number of tokens is an estimate based on the number of words. For this purpose, we trained a BPE tokenizer with vocabulary size 50,304 on our Wikipedia data source, tailoring the tokenizer for Czech. After training, we tokenized approximately a billion words from a variety of data sources, finding that the tokenizer split, on average, one word into 1.67 tokens. Hence, we compute the number of tokens by multiplying the reported word count by the constant 1.67.

Data source	# words	# tokens
Common Crawl HPLT CulturaX	91.95	153.56
MLP Books	0.15	0.25
iDnes	0.66	1.1
Wikipedia	0.18	0.3
Czech Sociological Review	0.003	0.005
Patents	0.55	0.92
Theses	0.59	0.99
SYN v9	5.67	9.47
Plenary Speeches	0.09	0.14
Total	99.84	166.73

Table 5.6. Final data size of the Czech Foundational Large Language Model Corpus. All numbers reported are in billions of words/tokens and rounded to two or three decimal places.

Chapter 6

Model Training

In this chapter, we will explore the potential of using our collected Czech corpus to train a large language model using the Hugging Face Transformers [56] library. This includes estimating the GPU hours necessary for training, as well as discussing the impact of both the quantity and quality of data on the performance of a language model.

Finally, we will present the training of several models and demonstrate their capabilities.

6.1 Training Time Estimates

Training LLMs on billions of tokens is a very computationally expensive and time demanding process. Thankfully, due to the parallel nature of the Transformer architecture, the training time can be reduced by distributing the process across multiple GPUs. In practice, the models recently released by companies, such as OpenAI or Meta, are trained on custom-built clusters with thousands of GPUs.

In this section, we estimate the computational resources, in terms of GPU hours, needed to train Transformer language models of various sizes with our dataset. To do so, we use the estimate equation for end-to-end training time provided in [57]. End-to-end training includes all operations including data loading, optimizer steps, communication and logging. The authors compute the time estimates as an approximation based on the model's number of parameters, the number of floating-point operations and the empirical throughput of an NVIDIA A100 GPU, which is a popular choice for model training.

For a Transformer model with l layers, hidden size h , vocabulary size V and sequence length s , the number of parameters can be computed as:

$$P = 12lh^2 \left(1 + \frac{13}{12h} + \frac{V + s}{12lh} \right)$$

The authors also set a lower bound for the number of floating-point operations (FLOPs) per iteration, which they calculate based on the number of matrix multiplications in the transformer and logit layers:

$$F = 96Bslh^2 \left(1 + \frac{s}{6h} + \frac{V}{16lh} \right)$$

where B is the batch size.

The number of iterations during training with T tokens can be computed as:

$$I = \frac{T}{Bs}$$

Then, they observe that $s \ll 6h$, $(V + s) \ll 16lh$ and $V \ll 12lh$. However, these observations are generally valid only for LLMs with billions of parameters. By combining

the previous equations with these observations, the authors approximate the end-to-end training time of a Transformer model:

$$\text{end-to-end training} \approx \frac{8TP}{nX}$$

where n is the number of GPUs and X is the empirical throughput of one GPU.

In their mixed-precision training experiments with models ranging from 2 to 70 billion parameters, the authors consistently achieved approximately 45% of the peak theoretical throughput of the A100 GPU, i.e. $X = 140$ TFLOP/s out of the peak 312 TFLOP/s.

We have computed training time estimates, reported in days, for a model with 1 billion parameters trained on our whole corpus, varying the number of available GPUs. The results are summarized in Table 6.1.

# tokens	# parameters	# A100	# days
167	1	4	28
		8	14
		16	7

Table 6.1. Training time estimates. Number of tokens and parameters reported is in billions.

However, it is important to bear in mind that these are merely rough estimates of the actual training time.

6.2 Data Quantity, Quality and Model Performance

When training language models on large amounts of textual data, it is important to consider how the performance of these models scales with the increased model size and training data. Simultaneously, the two variables are also connected to the cost of compute necessary for training.

The term “scaling laws” refers to the relationships that allow researchers to identify the compute-optimal regime for training models that are both performant and sample efficient. This means that training can be stopped before reaching convergence, without significantly affecting the model’s performance. However, scaling laws are only empirical observations which are not supported by standard machine learning theory.

Kaplan scaling laws [58] were an important milestone when assessing the impact of model size, training dataset size and computational cost on Transformer models. They found that performance, in terms of loss, increases when both model and dataset size are increased, with diminishing returns on model performance when one of them is increased while the other is fixed.

Altogether, they determined that for a given amount of compute C , the number of parameters P and dataset size D should be scaled as:

$$P \propto C^{0.73}$$

$$D \propto C^{0.27}$$

Therefore, for a compute-optimal model, a $10\times$ increase in computational budget should correspond to a $5.37\times$ increase in model parameters and a $1.86\times$ increase in

training tokens. This knowledge was used when training GPT-3 [4], a 175 billion parameters model trained on 300 billion tokens – approximately 1.7 tokens per model parameter.

Chinchilla scaling laws [59], released 2 years later, argued that the LLMs released at the time, including GPT-3, were severely undertrained.

Their experiments differed from Kaplan et al.’s in two key aspects. First, they did not use a fixed number of training tokens and learning rate schedule for all models – they set the training schedule to approximately match the number of training tokens. Second, the majority of their experiments used models with more than 500 million parameters, with certain models having up to 16 billion parameters, while their predecessors mostly used models with less than 100 million parameters.

When answering the question of what is the compute-optimal trade-off between number of parameters and number of training tokens, they used three different approaches. All approaches relied on training models with a varying parameter count and dataset size, using their training curves to fit an estimator of how the models should scale. They conclude that despite their differences, all approaches suggest that for an increased computational budget C , both model parameters P and training dataset size D should be scaled equally:

$$P \propto C^{0.5}$$

$$D \propto C^{0.5}$$

In other words, a tenfold increase in computational budget should correspond to a $3.16\times$ increase in model parameters and a $3.16\times$ increase in training tokens. The authors supported their scaling laws by training Chinchilla, a 70 billion parameter model trained on 1.4 trillion tokens – 20 tokens per parameter – which outperformed significantly larger models, including GPT-3, on several evaluation tasks.

To this day, the general validity of the previously presented scaling laws is uncertain. For example, in the recently released Llama 3 [60] family of models, the 8 billion parameter model has been trained on 15 trillion tokens, which is approximately $75\times$ more than suggested by Chinchilla scaling laws. Despite that, the model did not seem to converge. This fact hints that the currently available are severely undertrained and that the compute-optimal training regime which scaling laws propose can be overcome by smaller language models that are both performant and efficient during inference.

In TinyStories [61], the authors train small language models with sizes from 1 to 33 million parameters on a high-quality synthetic dataset of short stories for children. Despite their small size, the models achieve to generate diverse and consistent stories in coherent English, demonstrating emerging reasoning capabilities.

The work on TinyStories inspired another approach which focuses on quality, instead of quantity, of data. The Phi family of models [62–65] shows that even smaller models can match the performance of larger models, as long as they are trained on “textbook quality” data. The authors obtain such data by leveraging existing LLMs to filter internet content and create synthetic data. In this manner, they trained their latest 3.8 billion parameter model, phi-3-mini [65], on 3.3 trillion high-quality tokens, achieving comparable model performance to substantially larger models, such as Mixtral 8x7B [66] or GPT-3.5 Turbo (which is rumoured to have ~ 20 billion parameters).

6.3 Czech Models

In order to evaluate our corpus, we utilized it to train two Czech models of different sizes. As shown in Section 6.1, training a Transformer model with more than a billion parameters on our whole corpus requires multiple powerful (and expensive) GPUs, such as the NVIDIA A100, and multiple days of compute. For this reason, we have decided to train two significantly smaller models with only 70 and 160 million parameters.

The Pythia Scaling Suite [67] is a collection of English models designed for interpretability, available in different sizes and all trained on The Pile [47] dataset. Attempting to transfer some of the knowledge present in the pretrained Pythia models, we decided against training from scratch. Instead, we fine-tuned `pythia-70m-deduped`¹ and `pythia-160m-deduped`², two models trained on the deduplicated version of The Pile.

Furthermore, partially replicating the training data quality approach used in Phi [62], we trained our models only on a selection of data sources, which were deemed of high quality while being small in size, as compared to the rest:

- MLP Books
- iDnes
- Wikipedia
- Czech Sociological Review
- Patents
- Plenary Speeches

After shuffling the examples from the selected data sources, we employ the BPE tokenizer trained on Czech Wikipedia and first utilized in Section 5.3 to tokenize the selected data sources into more than 2 billion training tokens and 204,800 test tokens.

We trained the two models on the CIIRC cluster³ using 3 and 6 NVIDIA V100 GPUs for the 70 million and 160 million models, respectively. Training configuration details are available in Appendix A.

After training, we evaluate the perplexity of the two models on the test set. We also evaluate their checkpoints, saved once every 25% of training steps. The results are summarized in Table 6.2. As we can see, the test set perplexity of both models gradually decreases during training, achieving a reasonable perplexity. Furthermore, the larger model achieves a better result, as expected.

We have also generated some text examples using the 160 million parameter model, which can be found in Appendix B.

6.4 English-Czech Models

We also trained two bilingual models capable of generating both English and Czech text. To achieve this, we followed the steps outlined in [68], where the authors present a method to extend the capabilities of a pretrained model to a new language. Specifically, we used the 70 and 160 million parameter variants of the Pythia models and adapted them to Czech.

The first step is to adapt the tokenizer to efficiently encode both languages. As previously mentioned, the BPE tokenizer used in Pythia has a vocabulary of 50,304

¹ <https://huggingface.co/EleutherAI/pythia-70m-deduped>

² <https://huggingface.co/EleutherAI/pythia-160m-deduped>

³ <https://cluster.ciirc.cvut.cz>

Model	Checkpoint	Perplexity
Czech Model 70M	25%	53.61
	50%	38.51
	75%	34.1
	100%	31.85
Czech Model 160M	25%	59.42
	50%	35.46
	75%	28.85
	100%	26.09

Table 6.2. Test set perplexities of the Czech models. All numbers reported are rounded to two decimal places.

tokens. However, as it is trained for English, using it to tokenize Czech text creates significantly more tokens compared to an English corpus. This fact has direct implications for training and inference, making them take longer and requiring more computational resources. Additionally, using a suboptimal tokenizer negatively impacts model performance [69].

Therefore, we train a new BPE tokenizer with a vocabulary size of 5,000 – approximately 10% of the original, which the authors find to be an ideal compromise among various tested proportions. As before, we train the new tokenizer on Czech text from the Wikipedia data source. Then, we merge the original English-only tokenizer with the newly created Czech-only tokenizer by replacing the least common English tokens with the Czech ones. There are 1,938 overlapping tokens between the two tokenizers, which are left unchanged. We found that when tokenizing text in both English and Czech, the tokenizer split, on average, one word into 2.39 tokens.

We also reinitialize the weights of the model’s input embeddings at the positions of the replaced tokens using `torch.nn.init.trunc_normal_`, with standard deviation set to 0.02.

Neural networks trained sequentially on a new domain tend to lose their knowledge of their old domain [70]. This phenomenon naturally also applies to pretrained language models which, when trained on a new language, lose their ability to generate coherent text in the original language. To prevent this, we mix our Czech dataset from the previous section with English data from the deduplicated version of The Pile, which is the same dataset as Pythia was originally trained on.

Experiments with mixtures containing 25% and 50% of the original language data show that as long as some original language data is included, the adaptation process remains effective and is not highly sensitive to the exact proportion of the mixture [68]. Therefore, we chose to include over 300,000 words from the English dataset, which correspond to 25% of the Czech dataset with more than 1.3 billion words.

Similarly as for the Czech models, we shuffle the examples of our bilingual dataset and use our English-Czech tokenizer to create a training split with almost 4 billion tokens and a test split with 204,800 tokens.

As in the previous section, we trained the two models on the CIIRC cluster using NVIDIA V100 GPUs. Training configuration details are available in Appendix A.


We evaluate the perplexity of the two models on the test set as in the previous section. The results are summarized in Table 6.3. Similar to the Czech models from

the previous section, the test set perplexity of both models gradually decreases during training, with the larger model achieving better results. Overall, the perplexities reach very low values. It is important to note that the results are skewed towards the model’s performance in Czech, since English content forms only 25% of our dataset.

Model	Checkpoint	Perplexity
English-Czech Model 70M	25%	15.18
	50%	13.57
	75%	12.68
	100%	12.28
English-Czech Model 160M	25%	11.68
	50%	9.8
	75%	9.17
	100%	8.8

Table 6.3. Test set perplexities of the English-Czech models. All numbers reported are rounded to two decimal places.

As in the previous section, the text generation examples of the 160 million parameter model can be found in Appendix B.



Chapter 7

Conclusion

In this thesis, we have developed the largest Czech corpus for training large language models to date, consisting of 167 billion tokens. This corpus is composed of a variety of data sources, representing diverse writing styles ranging from webpages and news articles to academic papers and literary works. Such diversity ensures that the corpus captures all the linguistic nuances of the Czech language.

To ensure the quality of our corpus, the data was cleaned to remove any noise and irrelevant content, filtered to retain only the most relevant and high-quality text and deduplicated to eliminate redundancy. These steps were crucial in creating a robust dataset suitable for training state-of-the-art language models.

To demonstrate the capabilities of our corpus and to evaluate its effectiveness, we trained several language models on a subset of the data. The performance and text generation ability of these models highlight the potential of our corpus to serve as a robust base for training a Czech foundational large language model.

Future steps may involve extending our corpus with new data sources, including updating internet-based sources with fresh content. Additionally, more sophisticated cleaning and filtering procedures, such as those based on language models themselves, could be utilized to refine the corpus further. Moreover, the corpus could be enriched by incorporating synthetically generated data created by an existing LLM.

References

- [1] Daniel Jurafsky, and James H. Martin. *Speech and Language Processing (3rd Edition Draft)*.
- [2] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, and others. Improving language understanding by generative pre-training. 2018,
- [3] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language Models are Unsupervised Multitask Learners. 2019,
- [4] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. *Language Models are Few-Shot Learners*. 2020.
- [5] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivanjani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayanan Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. *PaLM: Scaling Language Modeling with Pathways*. 2022.
- [6] Gustav Šír. *SMU: Lecture 7 – Natural Language Processing 1*. 2023.
<https://drive.google.com/file/d/1EXwaraCccUibzDv3YHmiYTL6aNBf-JsQ/view>.
- [7] Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. *HellaSwag: Can a Machine Really Finish Your Sentence?* 2019.
- [8] Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. *Measuring Mathematical Problem Solving With the MATH Dataset*. 2021.
- [9] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. *Measuring Massive Multitask Language Understanding*. 2021.

- [10] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de OliveiraPinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. *Evaluating Large Language Models Trained on Code*. 2021.
- [11] Hugging Face. *Summary of the tokenizers*. ..
https://huggingface.co/docs/transformers/tokenizer_summary.
- [12] Rico Sennrich, Barry Haddow, and Alexandra Birch. *Neural Machine Translation of Rare Words with Subword Units*. In: Katrin Erk, and Noah A. Smith, eds. *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany: Association for Computational Linguistics, 2016. 1715–1725.
<https://aclanthology.org/P16-1162>.
- [13] Philip Gage. A new algorithm for data compression. *The C Users Journal archive*. 1994, 12 23-38.
- [14] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, ukasz Kaiser, and Illia Polosukhin. *Attention is All you Need*. In: I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, eds. *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2017.
https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.
- [15] Jeffrey L. Elman. Finding structure in time. *Cognitive Science*. 1990, 14 (2), 179-211. DOI [https://doi.org/10.1016/0364-0213\(90\)90002-E](https://doi.org/10.1016/0364-0213(90)90002-E).
- [16] Sepp Hochreiter, and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*. 1997, 9 (8), 1735-1780. DOI 10.1162/neco.1997.9.8.1735.
- [17] Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černocký, and Sanjeev Khudanpur. *Recurrent neural network based language model*. In: *Proceedings of the 11th Annual Conference of the International Speech Communication Association (INTERSPEECH 2010)*. International Speech Communication Association, 2010. 1045–1048. ISBN 978-1-61782-123-3.
<https://www.fit.vut.cz/research/publication/9362>.
- [18] Christopher Olah. *Understanding LSTM Networks*. 2015.
<https://colah.github.io/posts/2015-08-Understanding-LSTMs/>.
- [19] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. 2019.
- [20] . *JSON Lines Documentation*. ..
<https://jsonlines.org>.


- [21] . *JSON Documentation*. ..
<https://www.json.org/json-en.html>.
- [22] Yann Collet, and Murray Kucherawy. *Zstandard Compression and the 'application/zstd' Media Type*. RFC 8878. Request for Comments. 2021.
<https://www.rfc-editor.org/info/rfc8878>.
- [23] . *Zstandard Documentation*. ..
<https://facebook.github.io/zstd/>.
- [24] . *Apache Spark Webpage*. ..
<https://spark.apache.org>.
- [25] Matei Zaharia, Mosharaf Chowdhury, Michael J. Franklin, Scott Shenker, and Ion Stoica. *Spark: cluster computing with working sets*. In: *Proceedings of the 2nd USENIX Conference on Hot Topics in Cloud Computing*. USA: USENIX Association, 2010. 10.
- [26] Mikko Aulamo, Nikolay Bogoychev, Shaoxiong Ji, Graeme Nail, Gema Ramirez-Sanchez, Jorg Tiedemann, Jelmer van der Linde, and Jaume Zaragoza. *HPLT: High Performance Language Technologies*. In: *Proceedings of the 24th Annual Conference of the European Association for Machine Translation*. Tampere, Finland: European Association for Machine Translation, 2023. 517–518.
<https://aclanthology.org/2023.eamt-1.61>.
- [27] Marta Bañón, Jaume Zaragoza-Bernabeu, Gema Ramírez-Sánchez, and Sergio Ortiz-Rojas. *FastSpell: the LangId Magic Spell*. 2024.
- [28] Ona de Gibert, Graeme Nail, Nikolay Arefyev, Marta Bañón, Jelmer vander Linde, Shaoxiong Ji, Jaume Zaragoza-Bernabeu, Mikko Aulamo, Gema Ramírez-Sánchez, Andrey Kutuzov, Sampo Pyysalo, Stephan Oepen, and Jörg Tiedemann. *A New Massive Multilingual Dataset for High-Performance Language Technologies*. 2024.
- [29] Thuat Nguyen, Chien Van Nguyen, Viet Dac Lai, Hieu Man, Nghia Trung Ngo, Franck Dernoncourt, Ryan A. Rossi, and Thien Huu Nguyen. *CulturaX: A Cleaned, Enormous, and Multilingual Dataset for Large Language Models in 167 Languages*. 2023.
- [30] Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. *mT5: A massively multilingual pre-trained text-to-text transformer*. 2021.
- [31] Pedro Javier Ortiz Suarez, Benoit Sagot, and Laurent Romary. *Asynchronous pipelines for processing huge corpora on medium to low resource infrastructures*. In: Piotr Bański, Adrien Barbaresi, Hanno Biber, Evelyn Breiteneder, Simon Clematide, Marc Kupietz, Harald Lungen, and Caroline Iliadi, eds. Mannheim: Leibniz-Institut für Deutsche Sprache, 2019. 9 – 16.
<http://nbn-resolving.de/urn:nbn:de:bsz:mh39-90215>.
- [32] Pedro Javier Ortiz Suarez, Laurent Romary, and Benoit Sagot. *A Monolingual Approach to Contextualized Word Embeddings for Mid-Resource Languages*. In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, 2020. 1703–1714.
<https://www.aclweb.org/anthology/2020.acl-main.156>.
- [33] Julien Abadji, Pedro Ortiz Suarez, Laurent Romary, and Benoit Sagot. Towards a Cleaner Document-Oriented Multilingual Crawled Corpus. *arXiv e-prints*. 2022, arXiv:2201.06642.

- [34] Jan A. Botha, Emily Pitler, Ji Ma, Anton Bakalov, Alex Salcianu, David Weiss, Ryan McDonald, and Slav Petrov. *Natural Language Processing with Small Feed-Forward Networks*. In: Martha Palmer, Rebecca Hwa, and Sebastian Riedel, eds. *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Copenhagen, Denmark: Association for Computational Linguistics, 2017. 2879–2885.
<https://aclanthology.org/D17-1309>.
- [35] Edouard Grave, Piotr Bojanowski, Prakhar Gupta, Armand Joulin, and Tomas Mikolov. *Learning Word Vectors for 157 Languages*. In: Nicoletta Calzolari, Khalid Choukri, Christopher Cieri, Thierry Declerck, Sara Goggi, Koiti Hasida, Hitoshi Isahara, Bente Maegaard, Joseph Mariani, H
lene Mazo, Asuncion Moreno, Jan Odiijk, Stelios Piperidis, and Takenobu Tokunaga, eds. *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*. Miyazaki, Japan: European Language Resources Association (ELRA), 2018.
<https://aclanthology.org/L18-1550>.
- [36] Julia Kreutzer, Isaac Caswell, Lisa Wang, Ahsan Wahab, Daan van Esch, Nasanbayar Ulzii-Orshikh, Allahsera Tapo, Nishant Subramani, Artem Sokolov, Claytone Sikasote, Monang Setyawan, Supheakmongkol Sarin, Sokhar Samb, Benoît Sagot, Clara Rivera, Annette Rios, Isabel Papadimitriou, Salomey Osei, Pedro Ortiz Suarez, Iroro Orife, Kelechi Ogueji, Andre Niyongabo Rubungo, Toan Q. Nguyen, Mathias Müller, André Müller, Shamsuddeen Hassan Muhammad, Nanda Muhammad, Ayanda Mnyakeni, Jamshidbek Mirzakhlov, Tapiwanashe Matangira, Colin Leong, Nze Lawson, Sneha Kudugunta, Yacine Jernite, Mathias Jenny, Orhan Firat, Bonaventure F. P. Dossou, Sakhile Dlamini, Nisansa de Silva, Sakine Çabuk Ballı, Stella Biderman, Alessia Battisti, Ahmed Baruwa, Ankur Bapna, Pallavi Baljekar, Israel Abebe Azime, Ayodele Awokoya, Duygu Ataman, Orevaoghene Ahia, Oghenefego Ahia, Sweta Agrawal, and Mofetoluwa Adeyemi. Quality at a Glance: An Audit of Web-Crawled Multilingual Datasets. *Transactions of the Association for Computational Linguistics*. 2022, 10 50-72. DOI 10.1162/tacl_a_00447.
- [37] Hugo Laurençon, Lucile Saulnier, Thomas Wang, Christopher Akiki, Albert Villanova del Moral, Teven Le Scao, Leandro Von Werra, Chenghao Mou, Eduardo González Ponferrada, Huu Nguyen, Jörg Frohberg, Mario Šaško, Quentin Lhoest, Angelina McMillan-Major, Gerard Dupont, Stella Biderman, Anna Rogers, Loubna Ben allal, Francesco De Toni, Giada Pistilli, Olivier Nguyen, Somaieh Nikpoor, Maraim Masoud, Pierre Colombo, Javier dela Rosa, Paulo Villegas, Tristan Thrush, Shayne Longpre, Sebastian Nagel, Leon Weber, Manuel Muñoz, Jian Zhu, Daniel Van Strien, Zaid Alyafeai, Khalid Almubarak, Minh Chien Vu, Itziar Gonzalez-Dios, Aitor Soroa, Kyle Lo, Manan Dey, Pedro Ortiz Suarez, Aaron Gokaslan, Shamik Bose, David Adelani, Long Phan, Hieu Tran, Ian Yu, Suhas Pai, Jenny Chim, Violette Lepercq, Suzana Ilic, Margaret Mitchell, Sasha Alexandra Luccioni, and Yacine Jernite. *The BigScience ROOTS Corpus: A 1.6TB Composite Multilingual Dataset*. 2023.
- [38] Kenneth Heafield. *KenLM: Faster and Smaller Language Model Queries*. In: *Proceedings of the Sixth Workshop on Statistical Machine Translation*. Edinburgh, Scotland: Association for Computational Linguistics, 2011. 187–197.
<https://www.aclweb.org/anthology/W11-2123>.

- [39] Frederik Michel Dekking, Cornelis Kraaikamp, Hendrik Paul Lopuhaä, and Ludolf Erwin Meester. *A Modern Introduction to Probability and Statistics*. ISBN 978-1-85233-896-1.
- [40] Adrien Barbaresi. *Trafilatura: A Web Scraping Library and Command-Line Tool for Text Discovery and Extraction*. In: *Proceedings of the Joint Conference of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: System Demonstrations*. Association for Computational Linguistics, 2021. 122–131. <https://aclanthology.org/2021.acl-demo.15>.
- [41] Giuseppe Attardi. *WikiExtractor*. <https://github.com/attardi/wikiextractor>. 2015.
- [42] Radim Hladik. *Czech Sociological Review 1993-2016*. 2018. <http://hdl.handle.net/11372/LRT-2703>. LINDAT/CLARIAH-CZ digital library at the Institute of Formal and Applied Linguistics (FAL), Faculty of Mathematics and Physics, Charles University.
- [43] . *Czech Sociological Review Website*. .. <https://sreview.soc.cas.cz>.
- [44] Michal Kren, Vaclav Cvrcek, Jan Henys, Milena Hnatkova, Tomas Jelinek, Jan Kocek, Dominika Kovarikova, Jan Krivan, Jiri Milicka, Vladimir Petkevici, Pavel Prochazka, Hana Skoumalova, Jana Sindlerova, and Michal Skrabal. *SYN v9: large corpus of written Czech*. 2021. <http://hdl.handle.net/11234/1-4635>. LINDAT/CLARIAH-CZ digital library at the Institute of Formal and Applied Linguistics (UFAL), Faculty of Mathematics and Physics, Charles University.
- [45] Katherine Lee, Daphne Ippolito, Andrew Nystrom, Chiyuan Zhang, Douglas Eck, Chris Callison-Burch, and Nicholas Carlini. *Deduplicating Training Data Makes Language Models Better*. In: Smaranda Muresan, Preslav Nakov, and Aline Villavicencio, eds. *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Dublin, Ireland: Association for Computational Linguistics, 2022. 8424–8445. <https://aclanthology.org/2022.acl-long.577>.
- [46] Danny Hernandez, Tom B. Brown, Tom Conerly, Nova Dassarma, Dawn Drain, Sheer El-Showk, Nelson Elhage, Zac Hatfield-Dodds, Tom Henighan, Tristan Hume, Scott Johnston, Benjamin Mann, Christopher Olah, Catherine Olsson, Dario Amodei, Nicholas Joseph, Jared Kaplan, and Sam McCandlish. Scaling Laws and Interpretability of Learning from Repeated Data. *ArXiv*. 2022, abs/2205.10487
- [47] Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. The Pile: An 800GB Dataset of Diverse Text for Language Modeling. *arXiv preprint arXiv:2101.00027*. 2020,
- [48] Quentin Lhoest, Albert Villanova del Moral, Yacine Jernite, Abhishek Thakur, Patrick von Platen, Suraj Patil, Julien Chaumond, Mariama Drame, Julien Plu, Lewis Tunstall, Joe Davison, Mario Sasko, Gunjan Chhablani, Bhavitvya Malik, Simon Brandeis, Teven Le Scao, Victor Sanh, Canwen Xu, Nicolas Patry, Angelina McMillan-Major, Philipp Schmid, Sylvain Gugger, Clement Delangue, Theo Matussiere, Lysandre Debut, Stas Bekman, Pierric Cistac, Thibault Goehringer, Victor Mustar, Francois Lagunas, Alexander Rush, and Thomas Wolf. *Datasets:*

- A Community Library for Natural Language Processing*. In: *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, 2021. 175–184.
<https://aclanthology.org/2021.emnlp-demo.21>.
- [49] Robyn Speer. *ftfy*. Zenodo. 2019.
<https://doi.org/10.5281/zenodo.2591652>. Version 5.5.
- [50] Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. *The Curious Case of Neural Text Degeneration*. 2020.
- [51] Chenghao Mou, Chris Ha, Kenneth Enevoldsen, and Peiyuan Liu. *Chenghao-Mou/text-dedup: Reference Snapshot*. 2023.
<https://doi.org/10.5281/zenodo.8364980>.
- [52] Jure Leskovec, Anand Rajaraman, and Jeffrey David Ullman. *Mining of Massive Datasets*. 3 ed.. Cambridge University Press, 2020.
- [53] A.Z. Broder. *On the resemblance and containment of documents*. In: *Proceedings. Compression and Complexity of SEQUENCES 1997 (Cat. No.97TB100171)*. 1997. 21-29.
- [54] Chenghao Mou. *Large-scale Near-deduplication Behind BigCode*. 2023.
<https://huggingface.co/blog/dedup>.
- [55] Tibor Kiss, and Jan Strunk. Unsupervised Multilingual Sentence Boundary Detection. *Computational Linguistics*. 2006, 32 485-525. DOI 10.1162/coli.2006.32.4.485.
- [56] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. *HuggingFace’s Transformers: State-of-the-art Natural Language Processing*. 2020.
- [57] Deepak Narayanan, Mohammad Shoeybi, Jared Casper, Patrick LeGresley, Mostofa Patwary, Vijay Anand Korthikanti, Dmitri Vainbrand, Prethvi Kashinkunti, Julie Bernauer, Bryan Catanzaro, Amar Phanishayee, and Matei Zaharia. *Efficient Large-Scale Language Model Training on GPU Clusters Using Megatron-LM*. 2021.
- [58] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. *Scaling Laws for Neural Language Models*. 2020.
- [59] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de LasCasas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George vanden Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, and Laurent Sifre. *Training Compute-Optimal Large Language Models*. 2022.
- [60] AI@Meta. Llama 3 Model Card. 2024,
- [61] Ronen Eldan, and Yuanzhi Li. *TinyStories: How Small Can Language Models Be and Still Speak Coherent English?* 2023.
- [62] Suriya Gunasekar, Yi Zhang, Jyoti Aneja, Caio César Teodoro Mendes, Allie Del Giorno, Sivakanth Gopi, Mojan Javaheripi, Piero Kauffmann, Gustavo de Rosa,

- Olli Saarikivi, Adil Salim, Shital Shah, Harkirat Singh Behl, Xin Wang, Sébastien Bubeck, Ronen Eldan, Adam Tauman Kalai, Yin Tat Lee, and Yuanzhi Li. *Textbooks Are All You Need*. 2023.
- [63] Yuanzhi Li, Sébastien Bubeck, Ronen Eldan, Allie Del Giorno, Suriya Gunasekar, and Yin Tat Lee. *Textbooks Are All You Need II: phi-1.5 technical report*. 2023.
- [64] Microsoft Research Blog. *Phi-2: The surprising power of small language models*. 2023.
<https://www.microsoft.com/en-us/research/blog/phi-2-the-surprising-power-of-small-language-models>.
- [65] Marah Abdin, Sam Ade Jacobs, Ammar Ahmad Awan, Jyoti Aneja, Ahmed Awadallah, Hany Awadalla, Nguyen Bach, Amit Bahree, Arash Bakhtiari, Harkirat Behl, Alon Benhaim, Misha Bilenko, Johan Bjorck, Sébastien Bubeck, Martin Cai, Caio César Teodoro Mendes, Weizhu Chen, Vishrav Chaudhary, Parul Chopra, Allie Del Giorno, Gustavo de Rosa, Matthew Dixon, Ronen Eldan, Dan Iter, Amit Garg, Abhishek Goswami, Suriya Gunasekar, Emman Haider, Junheng Hao, Russell J. Hewett, Jamie Huynh, Mojan Javaheripi, Xin Jin, Piero Kauffmann, Nikos Karampatziakis, Dongwoo Kim, Mahoud Khademi, Lev Kurilenko, James R. Lee, Yin Tat Lee, Yuanzhi Li, Chen Liang, Weishung Liu, Eric Lin, Zeqi Lin, Piyush Madan, Arindam Mitra, Hardik Modi, Anh Nguyen, Brandon Norick, Barun Patra, Daniel Perez-Becker, Thomas Portet, Reid Pryzant, Heyang Qin, Marko Radmilac, Corby Rosset, Sambudha Roy, Olatunji Ruwase, Olli Saarikivi, Amin Saied, Adil Salim, Michael Santacrose, Shital Shah, Ning Shang, Hiteshi Sharma, Xia Song, Masahiro Tanaka, Xin Wang, Rachel Ward, Guanhua Wang, Philipp Witte, Michael Wyatt, Can Xu, Jiahang Xu, Sonali Yadav, Fan Yang, Ziyi Yang, Donghan Yu, Chengruidong Zhang, Cyril Zhang, Jianwen Zhang, Li Lyna Zhang, Yi Zhang, Yue Zhang, Yunan Zhang, and Xiren Zhou. *Phi-3 Technical Report: A Highly Capable Language Model Locally on Your Phone*. 2024.
- [66] Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego delas Casas, Emma Bou Hanna, Florian Bressand, Gianna Lengyel, Guillaume Bour, Guillaume Lample, Léo Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, Szymon Antoniak, Teven Le Scao, Théophile Gervet, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. *Mixtral of Experts*. 2024.
- [67] Stella Biderman, Hailey Schoelkopf, Quentin Anthony, Herbie Bradley, Kyle O’Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, Aviya Skowron, Lintang Sutawika, and Oskar vander Wal. *Pythia: A Suite for Analyzing Large Language Models Across Training and Scaling*. 2023.
- [68] Zoltan Csaki, Pian Pawakapan, Urmish Thakker, and Qiantong Xu. *Efficiently Adapting Pretrained Language Models To New Languages*. 2023.
- [69] Phillip Rust, Jonas Pfeiffer, Ivan Vulic, Sebastian Ruder, and Iryna Gurevych. *How Good is Your Tokenizer? On the Monolingual Performance of Multilingual Language Models*. In: Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli, eds. *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Online: Association for Computational Linguistics,



2021. 3118–3135.

<https://aclanthology.org/2021.acl-long.243>.

- [70] Michael McCloskey, and Neal J. Cohen. *Catastrophic Interference in Connectionist Networks: The Sequential Learning Problem*. Psychology of Learning and Motivation. 1989.

<https://www.sciencedirect.com/science/article/pii/S0079742108605368>.

Appendix A

Training Configuration

The training configuration we employed differs for the two model sizes, 70 and 160 million parameters, that we used. However, it is independent of whether the model is the Czech-only or English-Czech variant. We attempted to make the configuration details as closely aligned as possible with the ones used in the original Pythia models. The training configuration is summarized in Table A.1.

Configuration	Model size	
	70M	160M
context length	2,048	2,048
num_train_epochs	1	1
per_device_train_batch_size	4	2
gradient_accumulation_steps	16	32
save_strategy	steps	steps
save_steps	0.25	0.25
learning_rate	1e-3	6e-4
lr_scheduler_type	constant	constant
weight_decay	0.01	0.01
adam_beta1	0.9	0.9
adam_beta2	0.95	0.95

Table A.1. Training configuration of models.

Appendix B

Text Generation Examples

We have used the 160 million Czech and English-Czech models to generate some text examples. The text was generated with the configuration described in Table B.2. The prompts used for generation are highlighted in *italics*.

Configuration	Value
max_length	150
temperature	0.5
repetition_penalty	1.6
do_sample	True

Table B.2. Text generation configuration of models.

B.1 Czech Model 160M

- *Jsem studentem a rád bych, aby se mi dostalo dobrého vzdělání. A protože jsem si vědom toho i svých znalostí v oboru pedagogiky nebo psychologie (smích), tak mohu říci – že mám na starosti všechny oblasti vzdělávání pedagogických pracovníků ve školství...*
- *Ve volném čase rád sportuji a mám radost, že jsem se dostal do reprezentace. Ale to je přece jen trochu jiná kapitola než mistrovství světa v roce 2004,*” říká Štancl na adresu svého svěřence: „Vždycky mě štve tenhle sport.“

B.2 English-Czech Model 160M

- *Once upon a time, there was a dog who liked eating snacks and was the first to eat. The dog had been born in 1852, and he learned that his mother was an orphan when he left for England. The dogs were bred by Charles Darwin as well as John Locke’s son Thomas (1815-1910). They also breed the dogs that grew up with their parents’ dogs. The dogs are very rare in Europe because they have large breeds of dogs which grow into them at least twice every year. They can be found in several countries around the world including Japan, Australia, India, Russia, China, and Korea.*
- *Praha, hlavní město Česka, je historické město. V roce 1792 byla v Praze postavena železniční stanice a na jejím místě vznikla také pošta pro zamýšlené cestující (tzv. „Předměstí“). V letech 1868–1869 zde byly vybudovány dvě samostatné budovy: dnešního Masarykova nádraží s přilehlou nádražní budovu se dvěma výpravními kolejemi („Karlovarská“, KČT), kde jsou umístěny i nové nástupiště do depozituru bývalého vojenského velitelství.*