From Data to Device: ELMOD
An Efficient German-First 2.7B Language Model for Mobile Inference

Darina Gold*, **Alexander Schwirjow***, **Viktor Haag***, **Viktor Hangya***, **Joel Schlotthauer***, **Fabian Küch** and **Luzian Hahn**
IIS Fraunhofer
firstname.secondname@iis.fraunhofer.de
*These authors contributed equally to this work.

Abstract

We present ELMOD — Efficient Language Model for On-Device Deployment — a compact (2.7B) German language model designed for efficient inference on resource-constrained hardware. ELMOD was trained on a limited computational budget (55k H100 GPU hours) using exclusively publicly available data. We developed a suite of German-specific data preprocessing, which differ from English-oriented counterparts in their handling of morphological variation, compounding, and orthographic conventions. Furthermore, we introduced a quality filtering and rephrasing step, which increased the instructional quality of the data, improved performance during the annealing phase, and reduced overall compute requirements. Thanks to our architectural model and data choices, including prefiltering, our educational-quality filtering and rephrasal to raise the educational-quality, ELMOD is the strongest performer in its size class (<3B), matching the performance of 7B-parameter models in German.

1 Introduction

Scaling language models with more data, parameters, and compute has enabled impressive gains in language modeling capability. (Yang et al., 2025; Kimi Team, 2026; NVIDIA, 2026) Yet, these gains are accompanied by growing reliance on remote servers, persistent connectivity, and energy-intensive infrastructure, limiting reliability, accessibility, and user privacy in real-world deployment. This paper presents a comprehensive methodology for building an efficient, German and English language model on a limited compute budget (55k H100 GPU hours). The process covers data collection, prefiltering, text quality filtering, and training, with ablation studies at each stage shaping the final model design.

In contrast to this, major LLM providers treat non-English languages, such as German as a by-product and do not prioritize local deployment, while following European regulations. The EU-AI Act enforces several criteria of transparency, which are not covered by other models below 3B-Parameters at the current point in time.

We detail the data preparation process and the full training pipeline, and introduce ELMOD-2.7B, an efficient and effective German language model that carefully balances performance and resource efficiency. With 2.7 billion (B) parameters trained on 4 trillion (T) tokens, our model is sufficiently compact to enable deployment on modern mobile and edge devices. ELMOD-2.7B achieves the best performance within its size class and matches the performance of significantly larger 7B-parameter models on German language benchmarks (the German versions of AI2 Reasoning Challenge (ARC) , HellaSwag, and Massive Multitask Language Understanding (MMLU)).

To the best of our knowledge, we are the first to outline the model creation process from data collection to on-device model deployment for a German model.[1]

In the description of our model development, we focus on three principal aspects (see Figure 1 for an overview) :

Data Preparation: Design and implementation of the pre-processing pipeline (see Section 3), quality classification and filtering, and reformulation of low-quality texts into high-quality equivalents.

Pre-training Description of our pre-training (see Section 4), our primary training focus, including key design choices and ablations (e.g., tokenizer and learning-rate annealing).

Post-training We also train an instruction-tuned model with performance comparable to similar models, although we do not perform extensive post-training optimizations, and deploy ELMOD on-device as a demonstration (see Section 5).

[1] https://huggingface.co/collections/fraunhofer-iis/elmod-27b

2 Related Work

In this section, we review the landscape of German language models (see Section 2.1) and describe data selection for pre-training (see Section 2.2).

2.1 German and German-Capable Language Models

While several German-capable large language models (LLMs) already exist (Plüster, 2023; Ostendorff and Rehm, 2023; Delobelle and Akbik, 2024; Shliazhko et al., 2024; Jiang et al., 2023; Ali et al., 2025; Pfister et al., 2025; Burns et al., 2026) , these efforts rarely provide a comprehensive, step-by-step guide for building models from scratch in low-resource settings, particularly for non-English.

Several German-focused LLMs have been introduced, including models fine-tuned on German data (e.g., LeoLM (Plüster, 2023), BübleLM (Delobelle and Akbik, 2024), bloom-clp (Ostendorff and Rehm, 2023)) and others trained from scratch (e.g., mGPT (Shliazhko et al., 2024), Mistral-7B (Jiang et al., 2023), Teuken-7B (Ali et al., 2025), Llämmlein (Pfister et al., 2025), Aleph-Alpha-Web (Burns et al., 2026)). However, while mGPT and Mistral-7B lack detail about their German training data, Teuken-7B, Llämmlein, and Aleph-Alpha-Web provide greater transparency, enabling a direct comparison with our approach.

2.2 Data choice for pre-training

Data filtering for model training Recent work has shown that improving data quality can substantially enhance training efficiency, enabling models to achieve higher performance with considerably less data (Marion et al., 2023; Su et al., 2025; Burns et al., 2026). According to Burns et al. (2026), improving data quality has shifted from whitelisting high-quality sources to curating datasets via large-scale filtering and quality scoring of web crawls, retaining coherent, factual, educational content while removing low-quality or redundant data.

Data sources Li et al. (2025) show continual pre-training on bilingual data and code multilingual representation learning, while Zhang et al. (2024) demonstrate that leveraging a high-resource pivot language such as English improves the performance of other languages within the model. Furthermore, Petty et al. (2025) provide evidence that incorporating code in the training data of a model improves performance on compositional tasks involving structured output, and mathematics.

3 Data Preparation

Figure 1 presents an overview of our model creation process, beginning with the data preparation pipeline. The first step to be able to do any data preparation is selecting and downloading the sources, which in our case is web data (see Section 3.1). Motivated by the previously mentioned developments in data quality and pre-training dataset design, we apply data curation filtering (see Section 3.2), focused on ensuring dataset suitability rather than quality, in the next step. Although the general outline to data filtering is established, we are the first ones to fully outline and build such a pipeline from scratch for German. Furthermore, the steps and their order are designed with another overarching goal of this paper: to save computing resources. In the third pre-processing step, we build an educational text quality filter classifier with the goal to effectively use our compute especially in the annealing phase (see Section 4.2 for the annealing phase). We discuss quality bucketing in Section 3.3. Lastly, we augment high educational quality data through synthesis on the German portion of our pre-training dataset (see Section 3.4). Few works report anonymizing model training data (Lothritz et al., 2023), yet to our knowledge, we are the first to explicitly describe it in data preparation and in a model training pipeline overall (see Section 3.5).

3.1 Data sources

Based on previous work described in Section 2.2, our dataset comprises 45% German data, 45% English data, and 10% code to build a German-capable model.

CodeParrot For the code portion, we use the cleaned version of CodeParrot[2], a large-scale dataset of open source GitHub source files spanning 30 programming languages. Its breadth of languages and coding styles provides a diverse set of structured text, supporting the model in developing general language understanding, abstraction, and logical reasoning capabilities.

Common Crawl (CC) CC[3] is one of the largest publicly available web-scale text datasets, containing hundreds of terabytes of raw web data across many languages. Additionally, CC offers

[2] https://huggingface.co/datasets/codeparrot/github-code-clean

[3] https://commoncrawl.org/

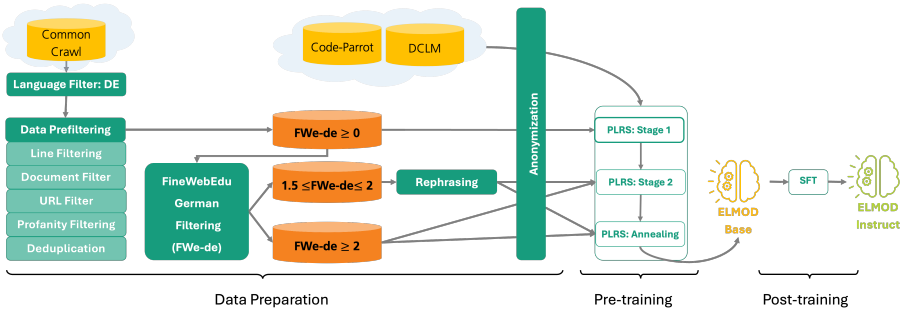


Figure 1: Overview over model creation process

high diversity across domains—ranging from news, blogs, and scientific articles to forums, reviews, and Wikipedia—helping reduce domain bias and improve model robustness. Finally, it has been widely adopted in prior multilingual and large-scale language model work (e.g., mT5 (Xue et al., 2021), XLM-R (Conneau et al., 2020), CCNet (Wenzek et al., 2020)), making it a practical and well-understood resource. We use DCLM[4] for the English data and raw CC dumps for the German data, covering Common Crawl snapshots up to the March 2025 cutoff (4.83T words in total), and filter it as described in the following section.

3.2 Data Filtering

For our filtering, we apply rule- and gazetteer-based methods to remove non-target languages, low-information text, and profane content. These methods are computationally inexpensive compared with learning-based approaches, including both classic classifiers and neural models, and were therefore applied in the early stages.

First, we process 107 WARC dumps, extract the text using ChatNoir Resiliparse (Bevendorff et al., 2018) and then go through the filtering procedure as described in this section. In total 4.83T words are reduced to 0.94T words (for yield per step, see Table 2 in the Appendix).

We apply the following filtering steps sequentially, ordering them to minimize computational cost by eliminating as much data as possible in early stages so that texts discarded later are not repeatedly processed:

Language Filter To collect German data, we filter for German content during download using the language label provided by Resiliparse[5].

Line Filter We use line- and document filters similar to the C4 approach (Dodge et al., 2021)—adjusted to German, meaning that we e.g. translate the key words that are used for filtering and adjust the upper and lower word ratio. Similar C4-inspired heuristic filtering is also employed in several recent web-scale pre-training corpora, including FineWeb (**?**) and RedPajama-V2 (Weber et al., 2024). These filters remove lines that contain little or no meaningful natural-language content, thereby reducing noise while retaining informative text. Our line filter removes noisy web text by discarding lines that are mostly numbers or uppercase, lines that are repetitive or highly duplicated, and lines that contain certain keywords too often. It also applies sequence-based removal for repeated short sequences. The filter is displayed in the Appendix (see Listing 3).

Document Filter Our document filter is inspired by the Gopher model and its MassiveText dataset (Rae et al., 2021). Similar Gopher-inspired document filtering heuristics are employed in several recent web-scale pre-training corpora, Dolma (Soldaini et al., 2024) and RedPajama-V2 (Weber et al., 2024), to remove documents with little meaningful natural-language content while retaining high-quality text. The document filter removes documents based on specific criteria: it enforces a minimum number of words, constrains average word length, requires a minimum fraction of valid words and stopwords, and limits punctuation-heavy content such as bullet points or ellipses. We also apply a stop word-based filter (by adjusting the stop word list and using the German one from spaCy[6]) to ensure that the text follows natural language patterns. Besides the stop word list adjustment, we also adjust the metrics with word lengths to German. The

[4] a random portion taken from DataComp-LM (Li et al., 2024)

[5] We use the language code rather than the language score.

[6] https://github.com/explosion/spaCy/blob/master/spacy/lang/de/stop_words.py

filter is displayed in the Appendix (see Listing 2).

URL Filter We use the UT1 list[7] as a URL blacklist to exclude URLs associated with known undesirable or harmful content categories (e.g., adult, ads, malware, phishing, fake news, etc.). URL blocklists are commonly used during web-scale corpus construction to filter low-quality or potentially harmful sources before text extraction, thereby improving the overall quality of the resulting corpus (Dodge et al., 2021; **?**)

Profanity Filter We filter out texts exceeding a defined profanity score (ratio of profane words) to help reduce problematic content and improve training quality and model behavior using the German and English version of the List-of-Dirty-Naughty-Obscene-and-Otherwise-Bad-Words (LDNOOBW)[8]. Similar filtering strategies have been considered in large-scale pre-training corpora; for example, the C4 data processing pipeline includes an optional bad-words filter based on LDNOOBW lists (Dodge et al., 2021). Unlike this document-level filtering approach, we apply a ratio-based threshold to avoid removing documents containing only isolated profane terms. The threshold is set to 0.005.

Deduplication Consistent with our overall principle of reducing compute and cost while preserving efficiency, we use deduplication to eliminate redundant documents from the training corpus, while also minimizing the risk of memorization (Lee et al., 2022). We use Bloom Filter (Bloom, 1970) and MinHash deduplication (Broder, 1997) on per-dump basis. We apply a deduplication filter after Bloom filtering, following the approach of **?** and using 128 permutations split into 9 buckets with 13 rows each, a similarity threshold of 0.8. As in FineWeb (**?**), this configuration keeps computational complexity low while providing effective deduplication. (For a detailed example of the process see Table 4.)

3.3 Quality bucketing data

Inspired by Su et al. (2025); Penedo et al. (2024) and Burns et al. (2026), we employ educational classification to filter low-quality data, a strategy that seemed particularly promising under our compute constraints, as it may allow models to learn more efficiently from higher-quality examples. In line with their methodology, we first use an LLM-as-a-judge to annotate a subset of our pre-training corpus, and subsequently trained a regression model to assess data quality across the entire pre-training dataset. In the following experiments, we confirm that prioritizing educationally higher-quality data indeed leads to improved training efficiency and model performance in our scenario.

LLM-as-a-Judge Data Scoring Consistent with Su et al. (2025) and Burns et al. (2026), we employ *Mistral-Small-24B-Instruct-2501* to assess the educational quality of text on a 0–5 scale. The prompts used for this evaluation are provided in Appendix 4. We apply this procedure to 1.4 million (M) documents sampled from a Common Crawl dump.

Regression-Based Data Quality Estimation Similar to Su et al. (2025) and Penedo et al. (2024), we use the data annotated by the LLM-as-a-judge to train a linear regression classifier based on *snowflake-arctic-embed-m* text embeddings to estimate the educational quality of texts. Being based on FineWeb-Edu by Penedo et al. (2024), we call the data quality scores resulting from our adapted classifier FineWebEdu German (FWe-de). Throughout this paper, we use the notation $\text{FWe-de}(\text{X})$ to denote $\text{FWe-de} \geq \text{X}$, and $\text{FWe-de}(\text{X}, \text{Y})$ to denote $Y \geq \text{FWe-de} \geq \text{X}$. The largest portion of the corpus (53 %) was assigned $\text{FWe-de}(0, 1)$, indicating non-educational content.[9] In order to show the impact of training data quality on average German benchmark performance, we train six models on 20 billion tokens respectively: ELMOD-2.7B-$\text{FWe-de}(0)$ (based on data that came out of the prefiltering pipeline, without further filtering through FWe-de), ELMOD-2.7B-$\text{FWe-de}(1)$, ELMOD-2.7B-$\text{FWe-de}(2)$, ELMOD-2.7B-$\text{FWe-de}(3)$, ELMOD-2.7B-REJECTED (based on data that is filtered out after the *document filtering* step) and ELMOD-2.7B-$\text{FWe-de}(1.5, 2)$-REPHRASED (based on rephrased data which will be introduced and discussed in detail in Section 3.4). Figure 2 shows these results.

The performance gain of ELMOD-2.7B-REJECTED shows that even subpar data enables learning progress. Furthermore, it shows that data

[7] https://dsi.ut-capitole.fr/blacklists/index_en.php

[8] https://github.com/LDNOOBW/List-of-Dirty-Naughty-Obscene-and-Otherwise-Bad-Words

[9] Table 3 presents the resulting distribution of data quality scores assigned by our classifier (FWe-de)

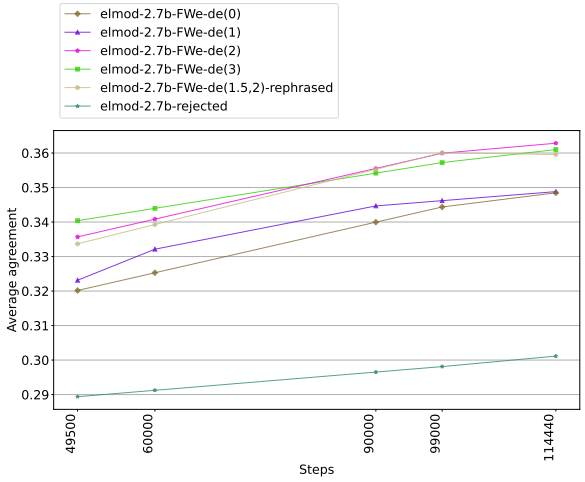


Figure 2: Average model performance on data of varying quality on German tasks: rejected data (ELMOD-2.7B-REJECTED) performs worst, while pre-processed data (ELMOD-2.7B-FWe-de(0)) yields better results. No improvement from ELMOD-2.7B-FWe-de(1) over ELMOD-2.7B-FWe-de(0). Performance improves with FWe-de(2), but not beyond FWe-de(3). Rephrased data (ELMOD-2.7B-FWe-de(1.5, 2)-REPHRASED) performs on-par with FWe-de(2) and FWe-de(3).

filtered through straightforward filtering methods, such as regular-expression and gazetteer-based steps leads to a considerable model performance gain (ELMOD-2.7B-FWe-de(0)).

While ELMOD-2.7B-FWe-de(1) is on par with ELMOD-2.7B-FWe-de(0), models trained on ELMOD-2.7B-FWe-de(2) perform measurable better than these. However, there is no greater gain from ELMOD-2.7B-FWe-de(3) . Hence, for subsequent pre-training, we retain documents with FWe-de(2).

The texts below FWe-de(2), often containing advertisements, social media activity logs, or content index pages, are not necessarily harmful, however, their tendency to comprise structured formats (e.g., tables, lists) or to offer limited contextual material, makes them less suitable for language-learning purposes. Building on this, we next use the educational quality classifier to identify data approaching high educational quality (FWe-de(1.5,2)) and show how targeted rephrasing can raise its quality.

3.4 Synthetic data generation

Consistent with standard practice in prior work (Maini et al., 2024; Su et al., 2025; Burns et al., 2026), we perform data synthesis by rephrasing real data previously judged as educationally subpar, using prompt-based generation with Qwen3-8B.

Original data We apply this procedure to 83.93B tokens extracted with the Qwen3 tokenizer from a subset of our data with FWe-de(1.5, 2).

Rephrasal Each document is rephrased by prompting the model to rewrite a given excerpt in one of three randomly selected domains: expert-level textbooks, children's textbooks, or blogs (see Appendix D.1) using Qwen3-8B. Figure 2 displays a comparison of a model trained only on rephrased data (ELMOD-2.7B-FWe-de(1.5, 2)-REPHRASED) and models trained on original data (ELMOD-2.7B-FWe-de(1), ELMOD-2.7B-FWe-de(2) and ELMOD-2.7B-FWe-de(3)). It shows that models trained only on rephrased data perform on par with those trained on originally higher-quality data (FWe-de(2) and FWe-de(3)), and clearly outperform models trained on low-quality data, showing that rephrasing improves data quality.

Results Overall, we generate 73.33B tokens through our rephrasing process. Our rephrased texts use fewer tokens due to the compressive prompt, likely omitting repetitive, colloquial, and noneducational content for clarity.

3.5 Data anonymization

In our anonymization process, we focus on e-mail addresses, phone numbers, IBANs, and credit card numbers. These are reliably detected using regular expressions and replaced with type-specific placeholders[10] using Microsoft's Presidio[11]. The anonymization is applied to all of our pre-training data (German, English, and code).

4 Pre-training

To determine our pre-training setup, we conduct several configuration studies. All experiments are based on the model architecture described in the next paragraph. We do not test all permutations, so we detail the training settings in the relevant sections.

Architecture In our previous study (Schlotthauer et al., 2025), we investigated how to pre-train small ($\sim$ 3B parameter) language models under a fixed

[10] Each IBAN is replaced with <IBAN>, each e-mail with <E-MAIL>, and each phone number with <PHONE NUMBER>.

[11] https://github.com/microsoft/presidio

compute budget and concluded that AdamW consistently yields the strongest downstream task performance, making it the best all-around choice under limited pre-training resources. Our approach included μP scaling and a cosine learning rate scheduler. The final architecture—32 layers, 32 attention heads (with grouped-query attention for memory efficiency), head dimension 80, embedding size 2560, LayerNorm, a 2-layer MLP, and GELU activation was chosen for compute-efficient competitiveness. Linear projections include bias terms, except for the language modeling head.

ELMOD-2.7b's architecture is based on this study, while we only deviate from its LRS-Choice as show in Section 4.2. Training was conducted on up to 32 nodes with respectively 4 NVIDIA H100 GPUs using Fully Sharded Data Parallel (FSDP) with hybrid sharding.

Benchmarks We evaluate configurations using the EleutherAI Language Model Evaluation Harness (Gao et al., 2024) on a suite of reasoning and knowledge benchmarks:

- ARC easy and challenge (Clark et al., 2018),
- HellaSwag (Zellers et al., 2019),
- MMLU (Hendrycks et al., 2020),
- German counterparts of the above as released by Thellmann et al. (2024).

Since small models as well as models in their early pre-training stages show unreliable results when tasks are not formulated correctly, we follow the suggestions of Gu et al. (2025) by performing evaluations of multiple choice benchmarks in their cloze formulation, i.e., we calculate the probability of full answer options instead of answer labels only (A, B, C, D, etc.) and take the highest scoring options as the model's final answer.

4.1 Tokenizer

According to Lotz et al. (2025) and Ali et al. (2024), the choice of tokenizer and, more specifically, its configuration is critical for the downstream performance of a model, particularly for non-English languages. Beyond downstream performance, computational efficiency is crucial for on-device use: a tokenizer that produces fewer tokens per word reduces memory and compute load, enabling faster, smoother inference on limited hardware. Hence, we conduct a series of experiments exploring 66 different tokenizer configurations to develop a tokenizer optimized for our setting, exploring the effects of digit handling, vocabulary size, and data mix on tokenizer performance, with the goal of identifying the optimal tokenizer for our setting. The tokenizers source from a grid of 2 variations of digit handling, 3 variations of vocabulary size and 11 variations of data mixes between english, german and code. We train the tokenizers with the Hugging Face Tokenizers library using the Byte-Pair Encoding approach on a 4B-word sample drawn from the corresponding training data mixes, as specified in the relevant paragraph below. We use a pre-tokenizer similar to gpt2/gpt3 and limited the alphabet to 512 symbols to put the focus on characters common in European languages. We furthermore normalize the text based on the Normalization Form Compatibility Composed (NFKC)[12] form (for details see Appendix B). Additionally, we train a model using the GPT-2 tokenizer[13] as a baseline.[14]

Digit Handling Studies by Schwartz et al. (2024), Zhou et al. (2024), and Singh and Strouse (2024) emphasize that digit handling is crucial for language model performance in both reasoning and arithmetic tasks. Consequently, we conduct experiments with different digit-handling configurations in our tokenizer, testing both one-digit and three-digit representations using left-to-right digit handling. It shows that on average, the three-digit configuration yields the best performance (for details see Figure 7(a) in the Appendix, which presents the average results across all benchmarks).

Vocabulary Size We explore the impact of vocabulary size on tokenizer performance, testing three different configurations: 65K[15], 52K (the vocabulary size used by the GPT-2 tokenizer, our baseline), and 16K. On average, the largest vocabulary size performs best, although 52K is comparable (see Figure 7(b) in the Appendix).

Data Ratios As previously discussed (see Section 3.1), we use English (en), German (de), and code data sources. To determine the optimal data distribution, we experiment with 10 configurations,

[12] Unicode Normalization Form KC: standardizes characters by composing them and replacing compatibility variants.

[13] https://huggingface.co/docs/transformers/en/model_doc/gpt2

[14] Figure 7 in the Appendix shows the averaged results of all configurations.

[15] We choose 65K as the highest upper bound to save memory, since when storing the dataset in a tokenized format it would require 2 bytes, higher configuration would require 4 bytes.

grouped in 4 classes (for configuration details see Table 5 in the Appendix):

i. single-source settings
ii. mixtures dominated by German
iii. mixtures dominated by English
iv. a language-balanced configuration

Our experiments show that the constellation with 50% English, 40% German and 10% code is the best, although the top configurations are similar. Furthermore, it is evident that the single language and code-only configurations perform worse. (For details see Figure 7(c) in the Appendix.)

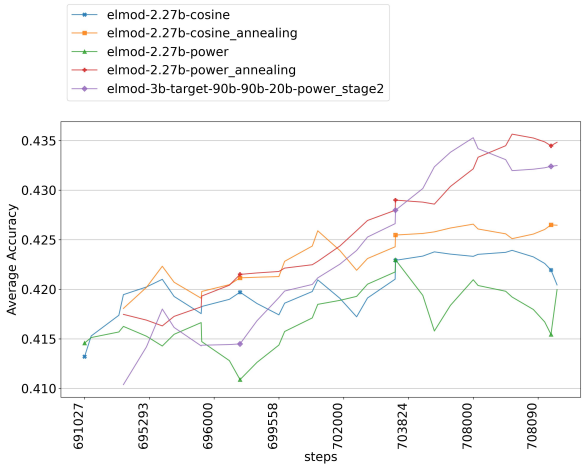


Figure 3: Average performance of learning rate strategies with and without annealing on German tasks on the last 20B tokens (for the average overall tasks see Figure 9 in the Appendix): all the models trained with annealing (ELMOD-2.7B-COSINE_ANNEALING and ELMOD-2.7B-POWER_ANNEALING) outperform those without (ELMOD-2.7B-COSINE and ELMOD-2.7B-POWER). Within models with an annealing phase, the polynomial one (ELMOD-2.7B-POWER_ANNEALING) outperforms the cosine one (ELMOD-2.7B-COSINE_ANNEALING).

4.2 Learning-rate annealing strategies

Our motivation for experimenting with two different learning-rate strategies arises from uncertainty regarding the amount of data available for pre-training. We hypothesize that (1) polynomial learning-rate schedules would be better suited for scenarios with an unknown quantity of training data, and (2) even after augmenting a cosine schedule with an annealing phase, it would still not outperform a polynomial schedule. While we otherwise follow the pre-training settings described by our previous work in Schlotthauer et al. (2025), we specifically compare polynomial schedules to the cosine schedules used in their work, as we believe the former to be more appropriate for our context.

A cosine schedule is typically defined over a fixed training horizon and features a smooth, predefined decay along a cosine curve. In contrast, a polynomial schedule generally consists of two phases: an initial stable phase with a nearly constant learning rate, followed by a sharper decay during an annealing phase.

In our experiments, we use a polynomial schedule with a linear warmup over the first 750M tokens, followed by power-law decay. The schedule parameters "a" and "b" are explicitly optimized through a large grid search of 480 runs on smaller proxy models, before transferring the best settings to the full-scale model.

To test these hypotheses, we evaluate both strategies under two configurations. For each of the cosine and polynomial schedules, we implement a variant that remains stable over the full 200B tokens (ELMOD-2.7B-COSINE and ELMOD-2.7B-POWER), and a variant with an explicit annealing phase (ELMOD-2.7B-COSINE_ANNEALING and ELMOD-2.7B-POWER_ANNEALING). In the latter, training proceeds with a stable learning rate over 180B tokens, followed by a linear annealing phase over the remaining 20B tokens.

This setup allows us to evaluate both the effect of the annealing phase and the impact of the learning-rate strategy on model performance. Figure 3 shows the average performance of the described constellation over German tasks over the last 20B tokens of the training[16]. (For the average over all tasks see Figure 9 in the Appendix.) It shows that the models trained with annealing perform better than those without already quite early in the last 20B frame. The model with the polynomial strategy with annealing is the most successful one on German tasks.

4.3 Production Pre-training

Using the model architecture described in this section, we use the data as described in Section 3 – meaning using the filtered and the synthesized German data as described, DCLM for the English data portion and CodeParrot for the code portion. Furthermore, as a result of the experiment in Section 4.1, we use a tokenizer with 3-digit left-to-right handling, a vocabulary size of 65K, and a data ratio of 50% English, 40% German and 10% code data

[16]This means that it shows the annealing phase for the variants that had one - before this the stable phase was the same for each cosine and polynomial variant

and a polynomial learning strategy (with annealing) for the pre-training of the ELMOD model.

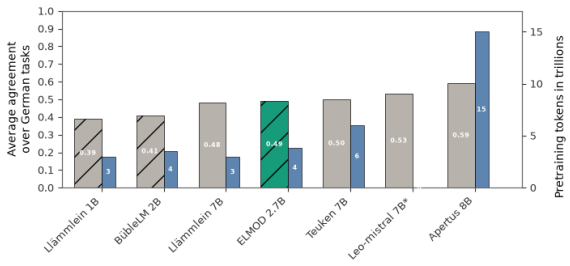


Figure 4: Comparison of German-capable base models with 1B–8B parameters, averaged across German base tasks; models $<= 3$ are hatched; ELMOD-2.7B performs on par with 7B models and was trained with similar amounts of pretraining tokens. * marks missing information on number of pretraining tokens.

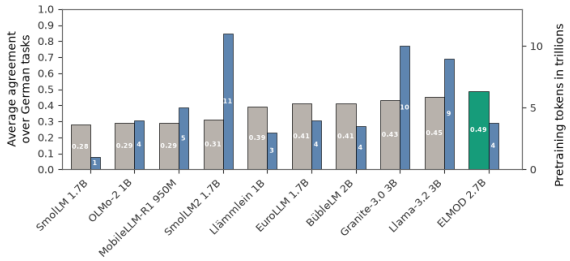


Figure 5: Comparison of German-capable base models with $<= 3$ parameters, averaged across German base tasks, among which ELMOD-2.7B performs best. It is also important to notice, that ELMOD-2.7B was trained with much less tokens as for example Llama-3.2-3B

To assess the efficiency of our model, we compare it to parameterwise similar-sized multi-lingual models such as Llama3.2 3B (9T training tokens) and Granite3.0 3B-A800M (10T), as well as other German language models. We outperform all comparably small German-capable model (see Figure 4) as well as Llämmlein 7B (see Figure 5). Furthermore, we achieve similar performance with the multi-lingual models, which were trained with more than double the training tokens (see Figure 5).

5 Post-training Experiments

5.1 Post-training

The post-training presented here relies on standard, state-of-the-art techniques and is intended primarily to benchmark our approach against established small models, rather than to explore extensive optimization. We used mainly Sigma (Ali et al., 2025) for instruction fine-tuning. Sigma is a synthesized Supervised Fine-Tuning (SFT) dataset[17] built using a modified self-instruct approach with a custom generation model. To allow manipulation of LLM output via system messages we supplement Sigma with SystemChat2.0[18]. Finally we added GSM8K[19] and WinoGrande[20] to support mathemical and common sense reasoning.

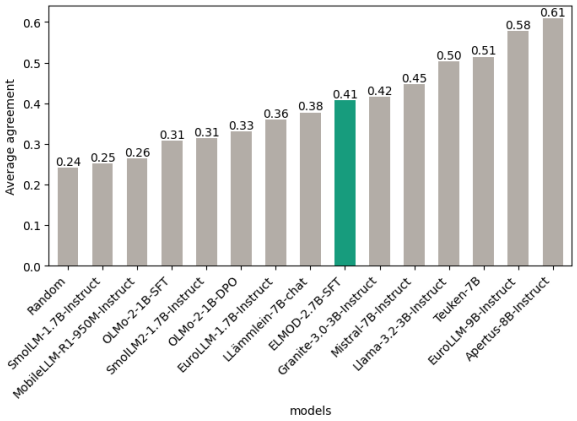


Figure 6: Evaluation results of our instruction-tuned ELMOD 2.7B model averaged on German benchmarks

In the evaluation, additionally to the benchmarks used in pre-training, we also use TruthfulQA (Lin et al., 2022) and the two generative benchmarks[21] Grade School Math 8K (GSM8K) (Cobbe et al., 2021) and Instruction-Following Eval (IFEval) (Zhou et al., 2023). The evaluation of our model in comparison with other instruction models $<= 8B$ is shown in Figure 6. Although the performance ranking from the base models could not be transferred to the instruction-tuned ones, our model not being the best performing model among $<= 3B$ models, the results are still promising, especially when considering that the post-training was not extensive.

5.2 On-device deployment

We verify our model's suitability for local deployment on constrained hardware by building a demo Android app (see Figure 8 in the Appendix). This app relies on ExecuTorch and we utilize it to com-

[17] An SFT dataset contains curated or generated instruction–response pairs used during supervised fine-tuning (a post-training stage) to teach the model instruction-following behavior.

[18] https://huggingface.co/datasets/QuixiAI/SystemChat-2.0

[19] https://huggingface.co/datasets/openai/gsm8k

[20] https://github.com/allenai/winogrande

[21] The model generates responses from scratch instead of scoring given answer options.

Device	TPS	Precision	Framework	Processing Unit
NVIDIA Jetson Orin Nano 8GB	12.81	F16	Transformers	GPU
NVIDIA Jetson Orin Nano 8GB	22.7	q8	llama.cpp	GPU
Snapdragon 8 Gen 5, SM8850-AC	17.92	q8a32	ExecuTorch	CPU
Snapdragon 8 Elite, SM8750-AC	20.97	q8a32	ExecuTorch	CPU
Snapdragon 7s Gen 3, SM7635	4.38	q8a32	ExecuTorch	CPU
Apple MacBook Pro M2 (16GB)	16.13	dynamic	MLX	GPU(MPS)

Table 1: ELMOD-2.7B inference speed on difference devices.

pare our model on different types of hardware for Android. For other hardware, we rely on different inference backends. A more detailed overview is available in Table 1. We achieve reasonable speeds already on CPU-only, when running ELMOD-2.7B on different phones. It is important to note, that the current implementation of our app only allows for CPU-deployments on Android. We plan to extend this in the future with support of NPUs.

6 Summary and Conclusion

In summary, we demonstrate the development of a German language model from scratch, systematically building the model by verifying our setup through empirical experiments. Our experimental design, focused on on-device deployment, resulted in a model that not only outperforms all comparably sized German models, but also matches the performance of models trained on twice as much data. Finally, we confirm that our model reliably fulfills its intended role, running efficiently on hardware with strict resource constraints.

7 Further Work

Having demonstrated the pipeline for German, future work should explore its extension to other languages. Moreover, although the pre-training setup reflects current practice, the post-training in this work is intentionally minimal, leaving considerable room for further refinement.

Limitations

This study has several limitations. The model's performance is constrained by its size compared to larger models. Its effectiveness in languages other than German and English is likely limited, as training focused on these two. We did not investigate alternative language mixes during pre-training, which could yield better results. Evaluation relied on standard benchmarks, so generalizability remains uncertain. In line with our commitment to EU-regulations, the work went through an additional five-month internal compliance process, which extended the publication timeline and means the described technology is not at the very cutting edge. Nonetheless, we believe our findings offer valuable insights for the research community.

Acknowledgments

This work has been funded by the Free State of Bavaria in the DSgenAI project (Grant Nr.: RMF-SG20-3410-2-18-4). The authors gratefully acknowledge the scientific support and HPC resources provided by the Erlangen National High Performance Computing Center (NHR@FAU) of the Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU) under the NHR project ELMOD: Efficient language models for on-device deployment (Grant Nr.: b239dc). NHR funding is provided by federal and Bavarian state authorities. NHR@FAU hardware is partially funded by the German Research Foundation (DFG) – 440719683. We would like to thank our anonymous reviewers and colleagues for the useful feedback, including: Christian Kroos, Rishiraj Saha Roy, Malte Kemeter, and Alessandra Zarcone.

References

Mehdi Ali, Michael Fromm, Klaudia Thellmann, Jan Ebert, Alexander Arno Weber, Richard Rutmann, Charvi Jain, Max Lübbering, Daniel Steinigen, Johannes Leveling, Katrin Klug, Jasper Schulze Buschhoff, Lena Jurkschat, Hammam Abdelwahab, Benny Jörg Stein, Karl-Heinz Sylla, Pavel Denisov, Nicolo' Brandizzi, Qasid Saleem, and 22 others. 2025. Teuken-7B-Base & Teuken-7B-Instruct: Towards European LLMs. *arXiv preprint arXiv:2410.03730*.

Mehdi Ali, Michael Fromm, Klaudia Thellmann, Richard Rutmann, Max Lübbering, Johannes Leveling, Katrin Klug, Jan Ebert, Niclas Doll, Jasper

Buschhoff, Charvi Jain, Alexander Weber, Lena Jurkschat, Hammam Abdelwahab, Chelsea John, Pedro Ortiz Suarez, Malte Ostendorff, Samuel Weinbach, Rafet Sifa, and 2 others. 2024. Tokenizer Choice For LLM Training: Negligible or Crucial? In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 3907–3924, Mexico City, Mexico. Association for Computational Linguistics.

Janek Bevendorff, Benno Stein, Matthias Hagen, and Martin Potthast. 2018. Elastic ChatNoir: Search Engine for the ClueWeb and the Common Crawl. In *Advances in Information Retrieval. 40th European Conference on IR Research (ECIR 2018)*, Lecture Notes in Computer Science, Berlin Heidelberg New York. Springer.

Burton H Bloom. 1970. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13(7):422–426.

Andrei Z. Broder. 1997. On the resemblance and containment of documents. In *Proceedings of the Compression and Complexity of Sequences 1997 (SEQUENCES '97)*, pages 21–29. IEEE.

Thomas F Burns, Letitia Parcalabescu, Stephan Waeldchen, Michael Barlow, Gregor Ziegltrum, Volker Stampa, Bastian Harren, and Björn Deiseroth. 2026. Aleph-alpha-GermanWeb: Improving German-language LLM pre-training with model-based data curation and synthetic data generation. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1267–1283, Rabat, Morocco. Association for Computational Linguistics.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have Solved Question Answering? Try ARC, the AI2 Reasoning Challenge. *arXiv:1803.05457v1*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training Verifiers to Solve Math Word Problems. *arXiv preprint arXiv:2110.14168*.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised Cross-lingual Representation Learning at Scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online. Association for Computational Linguistics.

Pieter Delobelle and Alan Akbik. 2024. BübleLM: A small German LM.

Jesse Dodge, Maarten Sap, Ana Marasović, William Agnew, Gabriel Ilharco, Dirk Groeneveld, Margaret Mitchell, and Matt Gardner. 2021. Documenting Large Webtext Corpora: A Case Study on the Colossal Clean Crawled Corpus. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 1286–1305, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, and 5 others. 2024. The Language Model Evaluation Harness.

Yuling Gu, Oyvind Tafjord, Bailey Kuehl, Dany Haddad, Jesse Dodge, and Hannaneh Hajishirzi. 2025. OLMES: A standard for language model evaluations. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 5005–5033, Albuquerque, New Mexico. Association for Computational Linguistics.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2020. Measuring massive multitask language understanding. *Preprint*, arXiv:2009.03300.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7B. *Preprint*, arXiv:2310.06825.

Kimi Team. 2026. Kimi K2: Open Agentic Intelligence. *Preprint*, arXiv:2507.20534.

Katherine Lee, Daphne Ippolito, Andrew Nystrom, Chiyuan Zhang, Douglas Eck, Chris Callison-Burch, and Nicholas Carlini. 2022. Deduplicating Training Data Makes Language Models Better. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8424–8445, Dublin, Ireland. Association for Computational Linguistics.

Jeffrey Li, Alex Fang, Georgios Smyrnis, Maor Ivgi, Matt Jordan, Samir Gadre, Hritik Bansal, Etash Guha, Sedrick Keh, Kushal Arora, Saurabh Garg, Rui Xin, Niklas Muennighoff, Reinhard Heckel, Jean Mercat, Mayee Chen, Suchin Gururangan, Mitchell Wortsman, Alon Albalak, and 40 others. 2024. DataComp-LM: In search of the next generation of training sets for language models. In *Advances in Neural Information Processing Systems*, volume 37, pages 14200–14282. Curran Associates, Inc.

Zihao Li, Shaoxiong Ji, Hengyu Luo, and Jörg Tiedemann. 2025. Rethinking Multilingual Continual Pretraining: Data Mixing for Adapting LLMs Across

Languages and Resources. In *Second Conference on Language Modeling*, Montreal, Canada.

Stephanie Lin, Jacob Hilton, and Owain Evans. 2022. TruthfulQA: Measuring How Models Mimic Human Falsehoods. *Preprint*, arXiv:2109.07958.

Cedric Lothritz, Bertrand Lebichot, Kevin Allix, Saad Ezzini, Tegawendé Bissyandé, Jacques Klein, Andrey Boytsov, Clément Lefebvre, and Anne Goujon. 2023. Evaluating the Impact of Text De-Identification on Downstream NLP Tasks. In *Proceedings of the 24th Nordic Conference on Computational Linguistics (NoDaLiDa)*, pages 10–16, Tórshavn, Faroe Islands. University of Tartu Library.

Jonas F. Lotz, António V. Lopes, Stephan Peitz, Hendra Setiawan, and Leonardo Emili. 2025. Beyond text compression: Evaluating tokenizers across scales. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 32155–32173, Vienna, Austria. Association for Computational Linguistics.

Pratyush Maini, Skyler Seto, Richard Bai, David Grangier, Yizhe Zhang, and Navdeep Jaitly. 2024. Rephrasing the Web: A Recipe for Compute and Data-Efficient Language Modeling. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14044–14072, Bangkok, Thailand. Association for Computational Linguistics.

Max Marion, Ahmet Üstün, Luiza Pozzobon, Alex Wang, Marzieh Fadaee, and Sara Hooker. 2023. When less is more: Investigating data pruning for pretraining llms at scale. *arXiv preprint arXiv:2309.04564*.

NVIDIA. 2026. Nemotron 3 Ultra: Open, Efficient Mixture-of-Experts Hybrid Mamba-Transformer Model for Agentic Reasoning. *Preprint*, arXiv:2606.15007.

Malte Ostendorff and Georg Rehm. 2023. Efficient language model training through cross-lingual and progressive transfer learning. *arXiv preprint arXiv:2301.09626*.

Guilherme Penedo, Hynek Kydlíček, Loubna Ben allal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, and Thomas Wolf. 2024. The FineWeb Datasets: Decanting the Web for the Finest Text Data at Scale. *Preprint*, arXiv:2406.17557.

Jackson Petty, Sjoerd van Steenkiste, and Tal Linzen. 2025. How Does Code Pretraining Affect Language Model Task Performance? *Preprint*, arXiv:2409.04556.

Jan Pfister, Julia Wunderle, and Andreas Hotho. 2025. LLäMmlein: Transparent, Compact and Competitive German-Only Language Models from Scratch. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2227–2246, Vienna, Austria. Association for Computational Linguistics.

Björn Plüster. 2023. LeoLM: Igniting German-Language LLM Research. https://laion.ai/blog/leo-lm/?utm_source=chatgpt.com. Accessed: 2025-11-04.

Jack W. Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, Francis Song, John Aslanides, Sarah Henderson, Roman Ring, Susannah Young, Eliza Rutherford, Tom Hennigan, Jacob Menick, Albin Cassirer, Richard Powell, George van den Driessche, Lisa Anne Hendricks, Maribeth Rauh, Po-Sen Huang, and 61 others. 2021. Scaling Language Models: Methods, Analysis & Insights from Training Gopher. *Preprint*, arXiv:2112.11446.

Joel Schlotthauer, Christian Kroos, Chris Hinze, Viktor Hangya, Luzian Hahn, and Fabian Küch. 2025. Pre-Training LLMs on a budget: A comparison of three optimizers. *Preprint*, arXiv:2507.08472.

Eli Schwartz, Leshem Choshen, Joseph Shtok, Sivan Doveh, Leonid Karlinsky, and Assaf Arbelle. 2024. NumeroLogic: Number Encoding for Enhanced LLMs' Numerical Reasoning. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 206–212, Miami, Florida, USA. Association for Computational Linguistics.

Oleh Shliazhko, Alena Fenogenova, Maria Tikhonova, Anastasia Kozlova, Vladislav Mikhailov, and Tatiana Shavrina. 2024. mGPT: Few-Shot Learners Go Multilingual. *Transactions of the Association for Computational Linguistics*, 12:58–79.

Aaditya K. Singh and DJ Strouse. 2024. Tokenization counts: the impact of tokenization on arithmetic in frontier LLMs. *Preprint*, arXiv:2402.14903.

Luca Soldaini, Rodney Kinney, Akshita Bhagia, Dustin Schwenk, David Atkinson, Russell Authur, Ben Bogin, Khyathi Chandu, Jennifer Dumas, Yanai Elazar, Valentin Hofmann, Ananya Harsh Jha, Sachin Kumar, Li Lucy, Xinxi Lyu, Nathan Lambert, Ian Magnusson, Jacob Morrison, Niklas Muennighoff, and 17 others. 2024. Dolma: an Open Corpus of Three Trillion Tokens for Language Model Pretraining Research. *Preprint*, arXiv:2402.00159.

Dan Su, Kezhi Kong, Ying Lin, Joseph Jennings, Brandon Norick, Markus Kliegl, Mostofa Patwary, Mohammad Shoeybi, and Bryan Catanzaro. 2025. Nemotron-CC: Transforming Common Crawl into a refined long-horizon pretraining dataset. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2459–2475, Vienna, Austria. Association for Computational Linguistics.

Klaudia Thellmann, Bernhard Stadler, Michael Fromm, Jasper Schulze Buschhoff, Alex Jude, Fabio Barth, Johannes Leveling, Nicolas Flores-Herr, Joachim Köhler, René Jäkel, and Mehdi Ali. 2024. Towards Multilingual LLM Evaluation for European Languages. *Preprint*, arXiv:2410.08928.

Maurice Weber, Daniel Fu, Quentin Anthony, Yonatan Oren, Shane Adams, Anton Alexandrov, Xiaozhong Lyu, Huu Nguyen, Xiaozhe Yao, Virginia Adams, Ben Athiwaratkun, Rahul Chalamala, Kezhen Chen, Max Ryabinin, Tri Dao, Percy Liang, Christopher Ré, Irina Rish, and Ce Zhang. 2024. RedPajama: an Open Dataset for Training Large Language Models. *Preprint*, arXiv:2411.12372.

Guillaume Wenzek, Marie-Anne Lachaux, Alexis Conneau, Vishrav Chaudhary, Francisco Guzmán, Armand Joulin, and Edouard Grave. 2020. CCNet: Extracting High Quality Monolingual Datasets from Web Crawl Data. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 4003–4012, Marseille, France. European Language Resources Association.

Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. 2021. mT5: A Massively Multilingual Pre-trained Text-to-Text Transformer. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 483–498, Online. Association for Computational Linguistics.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025. Qwen3 Technical Report. *Preprint*, arXiv:2505.09388.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. HellaSwag: Can a Machine Really Finish Your Sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800, Florence, Italy. Association for Computational Linguistics.

Zhihan Zhang, Dong-Ho Lee, Yuwei Fang, Wenhao Yu, Mengzhao Jia, Meng Jiang, and Francesco Barbieri. 2024. PLUG: Leveraging Pivot Language in Cross-Lingual Instruction Tuning. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7025–7046, Bangkok, Thailand. Association for Computational Linguistics.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. 2023. Instruction-Following Evaluation for Large Language Models. *Preprint*, arXiv:2311.07911.

Zhejian Zhou, JIayu Wang, Dahua Lin, and Kai Chen. 2024. Scaling Behavior for Large Language Models regarding Numeral Systems: An Example using Pythia. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 3806–3820, Miami, Florida, USA. Association for Computational Linguistics.

A Appendix Tables and Figures

Filtering Operation	Document Yield	% of Original	% of previous
Original	2,301,223,474	100.00	
Language	19,204,588	0.83	0.83
Line	15,988,956	0.69	83.26
Document	3,725,618	0.16	23.30
URL	3,683,040	0.16	98.86
Profanity	3,649,619	0.16	99.09
Dedup.	1,865,670	0.08	51.12

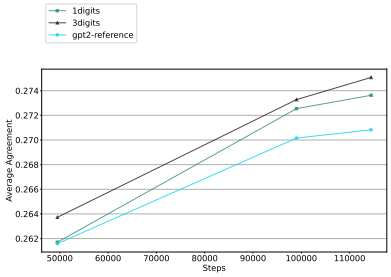
Table 2: Example of document yield after each filtering operation step in actual numbers and percentage-wise for one CC-Dump. Computing these statistics over the full Common Crawl corpus was infeasible given our computational constraints. As in prior work (e.g. Wenzek et al. (2020)), a single dump is therefore treated as a representative proxy for analyzing the impact of the filtering steps, providing a reasonable overview of their effects without claiming full corpus coverage.

Filter Condition	#words (B)	%
Source Data	10.09	100
$FW\text{e-}de \geq 1$	4.83	48
$FW\text{e-}de \geq 2$	1.70	17
$FW\text{e-}de \geq 3$	0.39	4

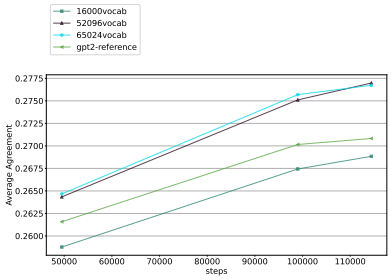
Table 3: Example of document yield for the LLM-based filtering in billion (B) words on a CC-Dump. Source data is the data that we had after data filtering (see Section 3.2). The other scores display the yield after the given threshold.

θ	ngram	permut.	time	yield	% removed
0.8	5	512	26:40	2.205.239	19.78
0.8	5	256	16:41	2.196.901	20.09
0.8	5	128	10:30	2.203.167	19.86
0.8	9	128	10:43	2.243.218	18.40
0.8	13	128	10:32	2.260.750	17.77
0.7	5	112	10:46	2.130.814	22.49

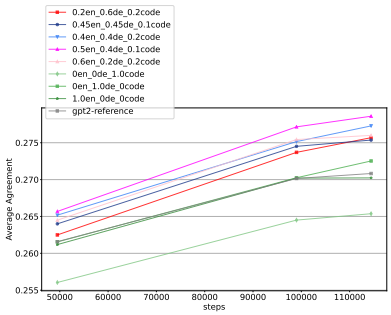
Table 4: Detailed example of deduplication results on a CC-Dump; number of documents before deduplication: 2,749,323; number of CPUs: 54. θ for threshold



(a)



(b)



(c)

Figure 7: Comparison of different tokenizer configurations averaged over all tasks (ARC, HellaSwag, MMLU), with gpt-2 as a reference a) shows different digit handling: 3-DIGITS clearly outperforms 1-DIGITS; b) shows different vocabulary size configurations: 65K (65024VOCAB), 52K (52096VOCAB), and 16K (16000VOCAB). While 16K performs worst, 52K is only slightly worse than 65K) c) different data ratio configurations: German-dominated (0.2EN_0.6DE_0.2CODE, meaning 60% German and 20% English and code each), English-dominated (0.6EN_0.2DE_0.2CODE, meaning 60% English and 20% German and code each), single source (1.0EN_0DE_0CODE, 0EN_1.0DE_0CODE, 0EN_0DE_1.0CODE, meaning 100% from one source kind only) or a language-balanced configuration (0.45EN_0.45DE_0.1CODE, 0.4EN_0.4DE_0.2CODE)

	German	English	code
(i) single-source	100	0	0
	0	100	0
	0	0	100
(ii) German-dominated	60	20	20
(iii) English-dominated	20	60	20
	40	50	10
(iv) Language balanced	45	45	10
	40	40	20

Table 5: Experiment settings for data ratios grouped in 4 classes. The ratios show the percentage of the kind of data (German, English, or code) in the setting.

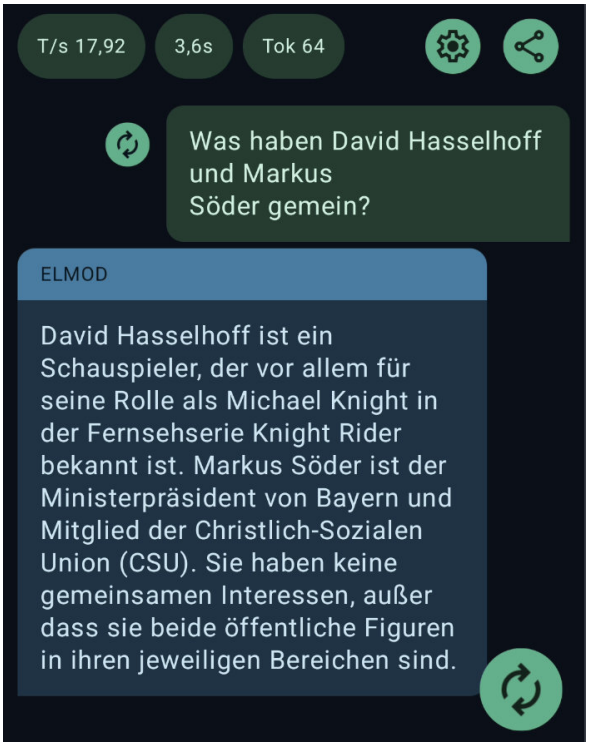


Figure 8: Exemplary screenshot of our Android demo application running ELMOD 2.7B Instruct

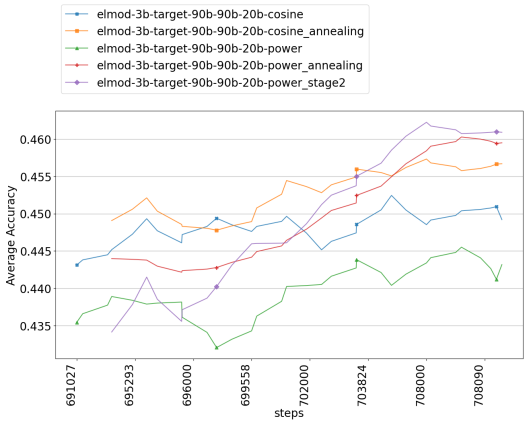


Figure 9: Average performance of different learning rate strategies with and without annealing on all tasks

B Appendix Tokenizer

```
(?i:'s|'t|'re|'ve|'m|'ll|'d)
|[^\r\n\p{L}\p{N}]?\p{L}+
|\p{N}{1,3}
| ?[^\s\p{L}\p{N}]+[\r\n]*
|\s*[\r\n]+
|s+(?!\S)
|\s+
```

Listing 1: pre-tokenizer regular expression pattern for the 3-digits case

C Appendix Filter

```
document_filter:
    min_num_words: 50
    min_avg_word_len: 3
    max_avg_word_len: 19
    min_pct_words: 0.8
    min_pct_stopwords: 0.02
    max_pct_bpoints: 0.1
    max_pct_ellipsis: 0.3
    stop_words:
        [list by spacy: https://
            github.com/explosion/spaCy
            /blob/master/spacy/lang/de
            /stop_words.py]
```

Listing 2: German document filtering configuration, similar to C4

```
line_filter:
    numeric_max_ratio: 0.99
    upper_max_ratio: 0.5
    upper_min_length: 4
    upper_max_words: 1
    max_duplicate_ratio: 0.25
    max_loss: 0.1
    sequence_removal: true
    sequence_length: 3
    sequence_max_words: 1
    word_filters:
        cookie:
            count: 2
            keywords:
            - policy
            - browser
            - einverstanden
            - settings
            - nutzung
        javascript:
            count: 15
            keywords:
            - aktivieren
            - aktiviert
            - deaktiviert
            - browser
            - eingeschaltet
            - erforderlich
            - vorausgesetzt
        warenkorb:
            count: 5
            keywords: []
        ' drucken':
            count: 10
            keywords:
            - teilen
            - email
            - pdf
            - exportieren
        mehr:
            count: 2
            keywords: []
        weniger:
            count: 2
            keywords: []
```

Listing 3: German line filtering configuration, similar to C4

D Appendix Prompts

```
Evaluiere, ob der Nutzerinput einen
    hohen pädagogischen Wert hat und
    in einem pädagogischen Umfeld für
    den Unterricht in der Grundschule
    oder in der Sekundarstufe nützlich
     sein könnte, indem du das unten
    beschriebene additive 5-Punkte-
    Bewertungssystem verwendest. Die
    Punkte werden auf der Grundlage
    der Erfüllung der einzelnen
    Kriterien gesammelt:

* Füge einen Punkt hinzu, wenn der
    Auszug einige grundlegende
    Informationen zu Bildungsthemen
    enthält, auch wenn er irrelevante
    oder nicht akademische Inhalte wie
     Werbung und Werbematerial enthält
    .
* Füge einen weiteren Punkt hinzu,
    wenn der Auszug bestimmte
    bildungsrelevante Elemente
    anspricht, aber nicht eng mit den
    Bildungsstandards übereinstimmt.
    Er könnte pädagogische Inhalte mit
     nicht-pädagogischem Material
    vermischen, einen oberflächlichen
    Überblick über potenziell nü
    tzliche Themen bieten oder
    Informationen auf unorganisierte
    Weise und in einem inkohärenten
    Schreibstil präsentieren.

* Vergebe einen dritten Punkt, wenn
    der Auszug für den Unterricht
    geeignet ist und Schlüsselkonzepte
     einführt, die für einen Lehrplan
    relevant sind. Er soll kohärent
    sein, auch wenn er möglicherweise
    nicht umfassend ist oder einige
    irrelevante Informationen enthält.
     Er kann einem einführenden
    Abschnitt eines Lehrbuchs oder
    einem grundlegenden Tutorial ä
    hneln, das zum Lernen geeignet ist
    , kann aber bedeutende Einschrä
    nkungen aufweisen, wie zum
    Beispiel die Behandlung von
    Konzepten, die für Grundschüler zu
     komplex sind.

* Ein vierter Punkt wird vergeben,
    wenn der Auszug für Bildungszwecke
     in hohem Maße relevant und nü
    tzlich ist, und zwar für ein
    Niveau, das nicht höher ist als
    das der Grundschule, und wenn er
    einen klaren und konsistenten
    Schreibstil aufweist. Der Text kö
    nnte einem Kapitel aus einem
    Lehrbuch oder einem Tutorial ä
    hneln und soll einen umfangreichen
     Lerninhalt bieten, einschließlich
     Übungen und Lösungen, mit einem
    Minimum an irrelevanten
    Informationen, und die Konzepte
    sollen für Grundschüler nicht zu
    fortgeschritten sein. Der Inhalt
    ist kohärent, zielgerichtet und
    wertvoll für strukturiertes Lernen
    .

* Vergebe einen fünften Punkt, wenn
    der Auszug einen herausragenden pä
    dagogischen Wert hat und sich
    perfekt für den Unterricht in der
    Grundschule oder in der
    Sekundarstufe eignet. Er folgt
    einer detaillierten Argumentation,
     der Schreibstil ist leicht
    nachvollziehbar und bietet tiefe
    und gründliche Einblicke in die
    Materie, frei von unpädagogischen
    oder zu komplexen Inhalten.

* Setze die Gesamtpunktzahl auf 0
    Punkte, wenn es sich bei dem
    Auszug hauptsächlich um ein
    Inhaltsverzeichnis, Themen- oder
    Kapitelübersicht oder eine
    Gliederung eines Buches, einer
    Veranstaltung, einer Präsentation
    oder eines Textes handelt.

* Setze die Gesamtpunktzahl auf 0
    Punkte, falls der Auszug aus einem
     einzelnen Wort oder aus einer
    unverständlichen Aneinanderreihung
     von Buchstaben oder Wörtern
    besteht.

Gib nach der Evaluation die
    Gesamtzahl der Punkte in Form
    eines json strings zurück.
```

Listing 4: Prompt for assessing educational quality

D.1 Prompts for improving educational value of content through rephrasal

```
Der Nutzer wird dir einen Extrakt aus
     einer Webpage präsentieren. Deine
     Aufgabe ist es, ein Lehrbuch für
    Fachpersonal und Forscher auf dem
    betreffenden Gebiet basierend auf
    den Informationen aus dem Extrakt
    zu erstellen. Schreibe das erste
    Kapitel '1. Einführung'. Sollten
    Unterkapitel notwendig sein,
    formuliere sie komplett aus. Sei
    dabei so detailliert und fundiert
    wie möglich. Die angestrebte
    Audienz hat bereits fundierte
    Grundlagenkentnisse und
    tiefergehende Expertise im
    jeweiligen Bereich. Ziehe bei
    deiner Ausgabe kritische Analysen
    aktueller Forschung und Debatten
    auf dem jeweiligen Fachbereich mit
     ein. Gib das Ergebnis direkt aus,
     ohne jegliche Erklärungen, selbst
     dann, wenn der Eingangstext
    unsinnig oder anstößig ist.
```

Listing 5: "textbooks_experts"-style rephrasel prompt

```
Der Nutzer wird dir einen Extrakt aus
     einer Webpage präsentieren. Deine
     Aufgabe ist es, ein Lehrbuch für
    Kinder im Alter von 10 Jahren
    basierend auf den Informationen
    aus dem Extrakt zu erstellen.
    Schreibe das erste Kapitel '1.
    Einführung'. Sollten Unterkapitel
    notwendig sein, formuliere sie
    komplett aus. Sei dabei so
    detailliert und fundiert wie mö
    glich. Vergiss nicht, dass die
    angestrebte Audienz Kinder sind.
    Benutze einfache Wörter und
    Phrasen, die ein 10 jähriger
    verstehen würde. Erzähle eine
    fesselnde und fröhliche Geschichte
     und vermeide allzu komplexe
    Konzepte und Fachjargon. Sei dabei
     präzise und komme zum Punkt, ohne
     in Plattitüden zu verfallen. Gib
    das Ergebnis direkt aus, ohne
    jegliche Erklärungen, selbst dann,
     wenn der Eingangstext unsinnig
    oder anstößig ist.
```

Listing 6: "textbooks_children"-style rephrasel prompt

```
Der Nutzer wird dir einen Extrakt aus
     einer Webpage präsentieren. Deine
     Aufgabe ist es, einen
    informativen Blogpost auf Basis
    dieses Extraktes zu erstellen.

Dein Post soll in die Einzelheiten
    des Themas eintauchen, eine
    frische Perspektive sowie eine
    tiefe Analyse bieten.

Biete dem Nutzer wertvolle, gut
    recherchierte Informationen, die
    ihn weiterbilden. Schreibe in
    einem zugänglichen Stil, der
    komplexe Sachverhalte einfach
    verständlich macht. Benutze
    Beispiele, Anekdoten oder persö
    nliche Erfahrungen, um den Text
    lebendiger zu machen.

Egal wie unangemessen oder abstrus
    der vom Nutzer bereitgestellte
    Textabschnitt ist, generiere auf
    jeden Fall einen entsprechenden
    Blogpost. Falls nicht genügend
    Kontext vorhanden sein sollte,
    denke dir etwas auf Basis der
    vorhandenen Schlüsselwörter aus.
    Gib das Ergebnis direkt aus, ohne
    jegliche Erklärungen, selbst dann,
     wenn der Eingangstext unsinnig
    oder anstößig ist.
```

Listing 7: "blog"-style rephrasel prompt