

Aleph-Alpha-GermanWeb: Improving German-Language LLM Pre-Training with Model-Based Data Curation and Synthetic Data Generation

Thomas F Burns

Letitia Parcalabescu

Stephan Wäldchen

Michael Barlow

Gregor Ziegler

Volker Stampa

Bastian Harren

Björn Deiseroth*

Aleph Alpha Research

Abstract

Scaling data quantity is essential for large language models (LLMs), yet recent findings show that data quality can significantly boost performance and training efficiency. We introduce a German-language dataset curation pipeline that combines heuristic and model-based filtering techniques with synthetic data generation. We use our pipeline to create Aleph-Alpha-GermanWeb, a 628B-word German pre-training dataset composed of three subsets drawing from: (1) Common Crawl web data (organic subset; 78B words), (2) FineWeb2 (organic subset; 235B), and (3) synthetically-generated data conditioned on actual, organic web data (synthetic subset; 329B). We evaluate our dataset by pre-training both a 1B Llama-style model and an 8B tokenizer-free hierarchical autoregressive transformer (HAT) from scratch. A comparison on German-language benchmarks, including MMMLU, shows significant performance gains of Aleph-Alpha-GermanWeb over FineWeb2 alone. This advantage holds at the 8B scale even when FineWeb2 is enriched by human-curated high-quality data sources such as Wikipedia. Our findings support the growing body of evidence that model-based data curation and synthetic data generation can significantly enhance LLM pre-training datasets.

1 Introduction

In recent years, ever-larger and more capable large language models (LLMs) have been released. This trend has led to the identification of power-law correlations between the loss value and the number of LLM parameters or pre-training dataset size (Hoffmann et al., 2022; Su et al., 2024b). As a result, researchers have argued (Hoffmann et al., 2022) that simply increasing the number of LLM parameters without also increasing the pre-training dataset

size yields diminishing returns. Additionally, considering the cost of LLM inference (Samsi et al., 2023) in production use-cases – which can far exceed the model’s initial training cost – practitioners may weigh the cost-benefit trade-off of replacing a larger model with a smaller one to minimise inference costs (Irugalbandara et al., 2024). To compensate for the reduced model size, a sufficiently-large training data budget would be required (De Vries, 2023; Hassid et al., 2024). However, this approach poses significant challenges for domains and languages where data is scarce.

Many recent examples in both pre-training (Marion et al., 2023; Su et al., 2024a) and post-training (Nguyen and He, 2025; Ye et al., 2025) of LLMs indicate that improvements to data quality can make training much more efficient. This is because higher-quality data can reduce the data quantity needed to reach or exceed the same model performance. In fact, conventional data scaling laws, which suggest that a small percentage increase in performance requires increasing the amount of training data by an order of magnitude, appear to be broken when data quality is improved (Sorscher et al., 2023). As a result, improving data quality allows us to train LLMs much more economically. Indeed, a recent 1.7 billion parameter language model pre-trained on 11 trillion tokens achieved new state-of-the-art results for its size by focussing heavily on data quality (Allal et al., 2025).

Curating datasets to improve their overall quality can be broadly grouped into four categories:

1. removing syntactic (near) duplication, *e.g.*, identical documents (or with slightly different wording), which are typically straightforward to identify and remove (Lee et al., 2022);
2. removing semantic duplication, *e.g.*, different summaries of the same book, which we expect to maintain some of for the purposes of syntactic variance, but should probably only keep

*Corresponding author bjoern.deiseroth@aleph-alpha-research.com

in proportion to the level of importance or complexity of the underlying concept (Abbas et al., 2023);

3. removing bad data, *e.g.*, code that does not compile (Di et al., 2024) or language which is ungrammatical (Penedo et al., 2024), less our models learn to repeat these mistakes; and
4. augmenting or upsampling good data, *e.g.*, adding high-quality synthetic data generated by LLMs (Long et al., 2024) or intentionally duplicating high-quality subsets of the data (Penedo et al., 2025).

Many curation methods originated as human-led and heuristic approaches, *e.g.*, preselecting sources with a reputation for high quality data or manual annotation and filtering of datasets (Penedo et al., 2023, 2024). More recently, however, model-based approaches, *e.g.*, training annotation models to classify data into different quality buckets (Li et al., 2024) or using LLM-based text quality filtering (Su et al., 2024a), have been shown to outperform heuristic methods. Additionally augmenting datasets with synthetically-generated data conditioned on high-quality organic data sources not only improves quantity but has been shown to further improve quality (Su et al., 2024a).

To date, model-based and synthetic approaches exist mostly for English-language text (Li et al., 2024; Su et al., 2024a). One of the most commonly-used and largest heuristically-filtered *non*-English web datasets is FineWeb2 (Penedo et al., 2025), which adapts prior heuristic methods shown to be successful for English web data (Penedo et al., 2023, 2024). Recently, model-based methods have been applied to a subset of languages from FineWeb2 to score them for quality, finding as little as 15% of the original dataset can be used to achieve comparable performance on 1 billion parameter LLMs (Messmer et al., 2025).

Nevertheless, there remains an overall quantity issue; FineWeb2, which includes data from over 1,000 languages, is <20% the size of the English-only FineWeb dataset (Penedo et al., 2024) (8TB vs. 47TB in total disk size). A single language like German constitutes <1.4% of FineWeb2, and model-based approaches only reduce this quantity further (Messmer et al., 2025). Meanwhile, synthetic approaches to improve the quantity, diversity, and quality of non-English language pre-training data have focussed on machine translation

(Wang et al., 2025), which comes with a host of socio-technical issues (Moorkens et al., 2024), and may lack the distinct syntactic or semantic advantages offered by synthetic approaches for LLM pre-training (Maini et al., 2024). Here we aim to fill these gaps by constructing a dataset for state-of-the-art German pre-training.

Contributions In this paper we present a data curation pipeline tailored to German-language LLM pre-training. We use it to derive our own dataset, Aleph-Alpha-GermanWeb, a 642B-word corpus composed of three complementary subsets drawn from three distinct sources: (1) Common Crawl (Foundation) web data not included in FineWeb2 (78B words); (2) FineWeb2 (ODC-By v1.0) (235B words); and (3) synthetic data conditioned on actual, organic web data (329B words). Our contributions are:

1. To curate the Common Crawl data, we applied a pipeline similar to (but which we show can perform better than) FineWeb2. We then augmented the dataset with synthetically-generated data and applied model-based quality classification methods;
2. We trained two different model architectures, a 1 billion parameter Llama-style model (Grattafiori et al., 2024) and a tokeniser-free 8 billion parameter model (Neitemeier et al., 2025b), on each subset of GermanWeb in isolation, without mixing data across subsets. For comparison, we trained the same architectures on FineWeb2 under identical training conditions. With these models, we evaluated a range of German language benchmarks, including Multilingual Massive Multitask Language Understanding (MMMLU) (Hendrycks et al., 2021). Our evaluations show that all three subsets of our dataset (synthetic, filtered Common Crawl, and our filtering of Fineweb2) outperform FineWeb2 (Penedo et al., 2025), even when the latter is combined with human-curated high-quality data sources such as Wikipedia at the 8 billion parameter scale. Our findings support the growing body of evidence that model-based data curation and synthetic data generation can significantly enhance pre-training datasets; and
3. We make GermanWeb publicly available to

the research community¹, in the hope that it will contribute to advancements in German-language LLM pre-training.

2 Methods

Our data curation methodology consists of three approaches: a data filtering pipeline for Common Crawl documents, synthetic document generation, and quality bucketing of all FineWeb2 documents.

2.1 Curating Common Crawl

We followed the RefinedWeb pipeline (Penedo et al., 2023), as adapted for German in FineWeb2 (Penedo et al., 2025), with some adjustments. We implemented all steps of the pipeline using NeMo Curator (Jennings et al.) (Apache-2.0). We summarise the number of web documents our pipeline yielded in Table 1.

Web data corpus. We downloaded six Common Crawl (Foundation) (CC BY-SA 4.0) web data dumps, spanning from September 2024 to February 2025. Each dump captured raw web page data, e.g., HTML, metadata such as URLs, and text extracts.

URL filtering. We used the URL filter developed in RefinedWeb (Penedo et al., 2023) to filter out: (i) fraudulent and adult websites from a blocklist of 4.6M domains; (ii) additional URLs based on the presence of words associated with fraudulent or adult content; and (iii) content from so-called ‘high-quality’ websites such as Wikipedia, arXiv, etc., which allows these datasets to be mixed in at a later point and in controlled proportions.

Text extraction. Natural language texts are embedded into raw web data via HTML and other web programming languages. Since we wish to focus on natural language, we extract the natural language from the raw web data using an extraction tool. RefinedWeb used the TRAFILATURA text extractor (Barbaresi, 2021). However, subsequent work (Li et al., 2024) constructing an English-language dataset from Common Crawl data showed that using the RESILIPARSE text extractor (Bevendorff et al., 2018) performed better on downstream LLM evaluations. We therefore use RESILIPARSE for text extraction.

Language identification. RefinedWeb (Penedo et al., 2023) used the fastText language classifier of

CCNet (Wenzek et al., 2020) at the document level. This classifier was trained on character n-grams from Wikipedia and supports 176 natural languages. Using this classifier, we removed documents for which the top language score was not German.

Repetition removal. Web-scraped documents can include repetitive phrases or words. Heuristic methods for detection and removal of these repetitions were notably developed by Rae et al. (2022). We used these methods with the repetition identification and removal thresholds set in FineWeb2 (Penedo et al., 2025) for German.

For example, we discarded the document if it consists of more than 28.2% duplicate lines. This example is shown in the first row of Table 2, where we also list the other repetition types we removed according to different thresholds for document length and document length measurement units.

Document-wise filtering. We used additional document-level heuristics introduced in Rae et al. (2022) and adapted in FineWeb2 (Penedo et al., 2025); namely, we required documents to have: (i) >50 and <100,000 words; (ii) a mean word length of <14 characters; (iii) a symbols-to-words ratio <0.1, where symbols are “#” and “...”; (iv) <90% of lines starting with a bullet point; (v) <30% of lines ending with ellipses; (vi) >77.4% of words containing at least one alphabetic character; and (vii) at least two of the following list of common German words: “DER”, “UND”, “DIE”, “IN”, “VON”, “IM”, “DEN”, “DES”, “MIT”, “DAS”, “ER”, “DEM”, “ALS”, “WURDE”, “FÜR” (English translations: “THE”, “AND”, “THE”, “IN”, “OF”, “IN”, “THE”, “OF”, “WITH”, “THAT”, “HE”, “THE”, “AS”, “WAS”, “FOR”).

Line-based filtering. Rae et al. (2022) also introduced line-wise corrections to replace or remove unwanted text within documents. We opted instead for a stricter setting by removing documents if: (i) >15% of the document consists of numbers; (ii) >50% of the lines have >50% uppercase characters; (iii) the average words per line is <10 words; or (iv) >40% of paragraphs contain boilerplate strings, e.g., “terms of use”, “privacy policy”, etc..

Exact deduplication. By first hashing the text of each document, we performed exact deduplication to remove all but one copy of identical documents, i.e., where there are two or more documents whose strings are equal in the dataset, we kept only one of these documents. We performed this deduplication

¹<https://huggingface.co/datasets/Aleph-Alpha/Aleph-Alpha-GermanWeb>

CC dump	2024-38	2024-42	2024-46	2024-51	2025-05	2025-08	Total
Original	1,893.59	1,659.31	1,774.51	1,745.08	2,026.56	1,780.59	10,879.64
URL + Lang.	114.14	99.38	107.58	107.29	127.27	108.78	664.43
Content	65.13	55.90	60.75	60.29	72.65	61.20	375.92
Exact Dedup.	45.39	38.05	41.40	47.35	49.65	48.02	269.87
Fuzzy Dedup.	44.53	37.38	40.66	40.07	48.74	42.25	253.64
Global Dedup.						Exact	168.19
Global Dedup.						Fuzzy	151.61

Table 1: Number of documents (in millions) throughout our curation pipeline across our downloaded Common Crawl data.

Repetition Type	Threshold	Metric
Duplicate lines	28.2%	# of lines
Duplicate paras	30.0%	# of lines
Duplicate paras	20.0%	# of paras
Repeated chars	20.0%	# of lines
Top 2-grams	7.7%	# of chars
Top 3-grams	10.1%	# of chars
Top 4-grams	12.3%	# of chars
Duplicate 5-grams	14.2%	# of chars
Duplicate 6-grams	12.7%	# of chars
Duplicate 7-grams	11.5%	# of chars
Duplicate 8-grams	10.6%	# of chars
Duplicate 9-grams	9.7%	# of chars
Duplicate 10-grams	8.8%	# of chars

Table 2: Repetition removal thresholds for methods introduced in Rae et al. (2022), with thresholds set as in FineWeb2 (Penedo et al., 2025) for German. In the table, ‘paras’ is short for ‘paragraphs’ and ‘chars’ is short ‘characters’.

procedure over the entire dataset. This approach is notably unlike FineWeb2 (Penedo et al., 2025), which retained a certain number of duplicate documents according to their duplication rate within the overall dataset, a technique they refer to as ‘rehydration’.

Fuzzy deduplication. Our document-level fuzzy deduplication methodology closely follows Smith et al. (2022), employing a multi-stage process to identify and remove near-duplicates within the corpus. The procedure began with computing Min-Hash signatures (Broder, 1997) using character-based 5-grams with 23-character sequences, where the number 23 was calculated by assuming an average of 4.5 characters per word. We then used locality sensitive hashing (LSH) (Gionis et al., 1999) to identify candidate duplicates and sort them into 14 buckets, which each had eight unique hashing func-

tions. LSH buckets were then converted to edges for the connected components algorithm. This algorithm identified document clusters with high similarity, producing groups of near-duplicate documents that can then be removed from the corpus. We performed this deduplication procedure over the entire dataset.

2.2 Synthetic data generation

Inspired by similar work for English-language web data (Maini et al., 2024; Su et al., 2024a), we performed synthetic data generation conditioned on actual – what we also refer to as ‘organic’ – data from FineWeb2. We showcase the five prompt templates we used for synthetic data generation in Appendix B, designed to generate educational or basic rephrasings, summarisations, question-answer pairs, and lists of factual information. For all synthetic generation, we used the 12 billion parameter Mistral-Nemo-Instruct-2407² LLM.

In all cases, if the entire document was longer than a specified character threshold, we segmented it into chunks of characters with lengths equal to or less than the threshold. This was done to prevent the organic document from potentially taking up all remaining context space in the prompt and to improve the quality of the LLM’s responses – prior studies (Maini et al., 2024; Su et al., 2024a) also noted for English-language that if too much organic text is inserted it degrades the synthetic data quality. To avoid splitting the documents mid-word, -sentence, or -paragraph, and thereby potentially giving impractical or nonsensical inputs to the LLM, we segmented the document texts using TEXT-SPLITTER³. For a given prompt and document, we synthesized data for all semantic chunks until the organic document’s data was depleted. For

²<https://huggingface.co/mistralai/Mistral-Nemo-Instruct-2407> (Apache 2.0)

³<https://github.com/benbrandt/text-splitter>

each synthesised output, we also performed some post-processing using regular expressions and filtering to remove what we call ‘LLM artifacts’, such as prefixes like “Here’s the rephrased version:” and repetitions of the prompt or input text.

For a large number of FineWeb2 documents, we synthesized data based on one or more of our prompts. This resulted in many documents, especially long ones (which were segmented into more chunks), being represented in multiple ways in the final synthetic dataset. As mentioned in [Su et al. \(2024a\)](#), there is a non-negative (albeit diminishing) performance gain seen with exactly the same data during multi-epoch pre-training ([Muennighoff et al., 2023](#)), therefore it is reasonable to expect that additional ‘synthetic epochs’ conditioned on the same underlying organic data is not harmful up to a similar limit. In the work of [Muennighoff et al. \(2023\)](#), this limit was shown to be approximately four epochs. We thus chose to limit our prompt templates to five, such that there would be no more than five possible synthetic outputs conditioned on the same organic input when the whole dataset was used for training.

2.3 Quality classification

Following prior work on quality classification for English-language web data ([Su et al., 2024a](#)), we categorised documents into five quality buckets – high, medium-high, medium, medium-low, and low – by leveraging the strengths of several imperfect quality classifiers to produce a more accurate outcome. Specifically, we created several individual quality classifiers based on fastText ([Bojanowski et al., 2016](#)) and Bidirectional Encoder Representations from Transformers (BERT) ([Devlin et al., 2019](#)) models. In the following, we describe (1) our two grammar classifiers, (2) our two educational quality classifiers, (3) our two instruction classifiers and (4) the way in which we ensembled all classifiers and grouped the data into the five quality buckets according to the classifiers’ scores.

2.3.1 Grammar

We used LanguageTool⁴ to annotate a random subset of 400,000 German FineWeb2 documents with the DE_AGREEMENT rule, which identifies text passages with grammatical disagreement. To train our classifiers, we randomly selected 75,000 documents without identified grammar mistakes as high quality examples. As low quality examples, we

⁴<https://dev.languagetool.org>

took 75,000 random documents containing at least one identified grammar error.

We trained a fastText binary classifier on 95% of the data to classify the high and low quality examples, using the remaining 5% for validation. The model reached 63% precision and 63% recall on the validation set. We further trained a BERT-based binary classifier on the same train-validation splits, reaching a precision of 67% and recall of 66% on the validation set.

2.3.2 Educational quality

Inspired by successful demonstrations for English with FineWeb ([Penedo et al., 2024](#)) and Nemotron-CC ([Su et al., 2024a](#)) (Apache 2.0), we created educational quality classifiers to curate high-quality German web data by training a classifier on scores given by an LLM-as-a-judge. The intuition behind this is to identify web documents that are more likely to be educational, informative, and useful for model training, as opposed to spam, low-quality, or otherwise unhelpful content.

We first use an LLM-as-a-judge to label a small subset of the data. We then leverage these labelled documents as training data to train more efficient and lightweight fastText and BERT classifiers. This two-stage approach enables us to score a large number of web documents in a computationally-efficient way, while at the same time introducing implicit regularisation to the judging process.

We again made use of Mistral-Nemo-Instruct-2407, this time to annotate a random set of 600,000 documents from German FineWeb2 according to three criteria: (1) content quality, assessing coherence, informativeness and overall quality of the content; (2) language quality, evaluating the use of language, including formality, objectivity, and the presence of errors or slang; and (3) orthography, assessing the correctness of grammar, spelling, and punctuation, including errors such as typos, incorrect verb conjugation, and incorrect declension. We show the prompt used for the LLM-as-a-judge in [Appendix A](#), which provided scores from one to five for each criterion.

For each document, we calculated a combined *educational quality score* by taking the minimum over the three criteria rated by the LLM-as-a-judge. This is because we observed that the judge would often miss poor quality in one criterium, but rarely in all three. We then used these scores as the training signal for fastText and BERT quality classification models. We trained a BERT model tasked to

Model	Condition	Points
–	Initialise all documents	0
BERT (edu. quality)	Predicted score = 5	+3
fastText (edu. quality)	Predicted “high quality” with >99% confidence	+2
BERT (grammar)	Predicted “high quality”	+3
fastText (grammar)	Predicted “high quality” with >99% confidence	+2
BERT (instruction)	High quality probability is in top 15% of the distribution	+6
fastText (instruction)	High quality probability is in top 15% of the distribution	+4

Table 3: Overall quality scoring rules by model, condition, and assigned points.

predict the scores given the first 512 tokens of the document’s text, and a binary fastText classifier.

For the binary fastText classifier, the low- and high-quality training data subsets consisted of 185,403 documents each. The low quality subset consisted of documents with educational quality scores of one or two, whereas the high quality subset consisted of documents scoring four or five. We used 95% of the data (and the remaining 5% for validation) to train a fastText model to classify between high- and low-quality data. It reached 77% precision and 77% recall on the validation set.

To train the BERT classifier, we randomly selected a maximum of 75,000 documents from each class, which we had previously labelled with the LLM-as-a-judge with scores ranging from one to five. The resulting dataset consisted of 75,000 documents for each score above one, and 25,981 documents with scores of one. We used 95% of this dataset for training to predict the one to five scores. The model achieved an overall accuracy of 42% and a macro-average accuracy of 46% when evaluated on the remaining 5% of the data, which served as the validation set.

2.3.3 Instruction

Building on Messmer et al. (2025), we trained binary classifiers using both fastText and BERT to distinguish between chat/instruction-style data (high quality) and other types of documents (low quality). To create a balanced training dataset, we created two subsets of 80,000 documents each. The low-quality subset consisted of randomly selected documents from the German FineWeb2 corpus, while the high-quality subset comprised randomly sampled instances from the Aya collection, Include Base-44, OpenAssistant2, and MMLU. To create samples from MMLU, we first concatenated a question and the respective answers. For OpenAssistant-

2, we concatenated all samples that belong to the same conversation. We used 95% of the data for training and the remaining 5% for validation.

The fastText classifier indicated strong discriminative performance, achieving 99% precision and 99% recall on the validation set. Similarly, the BERT-based classifier reached a validation accuracy of 100%.

2.3.4 Ensembling

To split the data into different quality buckets, we use the rules in Table 3 to allocate *overall quality* points to each document. Using these scores, we categorised documents into five quality buckets based on the overall quality, as shown in Table 4.

Bucket	Score	Quality Distribution	
		Docs	Bytes
High	≥ 12	12.1%	8.1%
Medium-high	9 – 11	15.0%	9.9%
Medium	5 – 8	36.3%	28.7%
Medium-low	3 – 4	15.5%	21.7%
Low	< 3	21.0%	31.6%

Table 4: FineWeb2 document quality classifications decided by overall quality score.

3 Experiments

We trained and evaluated a one billion parameter (1B) Llama-style transformer model and an eight billion parameter (8B) hierarchical autoregressive transformer (HAT) model on all three subsets of our dataset in isolation to assess their quality compared to FineWeb2.

3.1 Evaluations

We measured model performance using the following German-language question-answering tasks.

MMMLU The Multilingual Massive Multitask Language Understanding dataset (Hendrycks et al., 2020), is a benchmark designed to evaluate the performance of LLMs across multiple languages and disciplines. It extends the original MMLU benchmark by translating its test set into 14 languages—including German—using professional human translators to ensure accuracy. The dataset encompasses 57 subjects ranging from elementary-level topics to advanced professional fields such as law, physics, history, and computer science. The German portion contains 14,000 samples.

The following three popular LLM benchmarks were translated from English to German (Plüster).

ARC The AI2 Reasoning Challenge (ARC) (Clark et al., 2018) is a benchmark dataset developed by the Allen Institute for AI to evaluate the reasoning capabilities of artificial intelligence models. It comprises 7,787 multiple-choice science questions sourced from standardized tests designed for students in grades three through nine. The dataset is divided into two subsets: ARC-Easy and ARC-Challenge; here we evaluate the translation of ARC-Easy.

HellaSwag The HellaSwag benchmark dataset (Zellers et al., 2019) is designed to evaluate the commonsense reasoning abilities of AI models, particularly in the context of sentence completion tasks. It comprises approximately 70,000 multiple-choice questions from diverse sources, including instructional videos and articles from platforms like WikiHow and ActivityNet. Each question presents a context followed by four possible sentence completions, one of which is correct.

TruthfulQA The TruthfulQA dataset (Lin et al., 2021) is a benchmark designed to evaluate the truthfulness of language models when generating answers to questions. It comprises 817 questions across 38 categories, including health, law, finance, and politics. The questions are crafted to challenge models with scenarios where humans might hold incorrect beliefs or misconceptions, aiming to assess whether models can avoid generating false answers learned from imitating human texts.

3.2 1B Llama-style model

We pretrained 1B Llama-style models on a total budget of approximately 84 billion tokens from different datasets. Training details can be found in Appendix C.1.

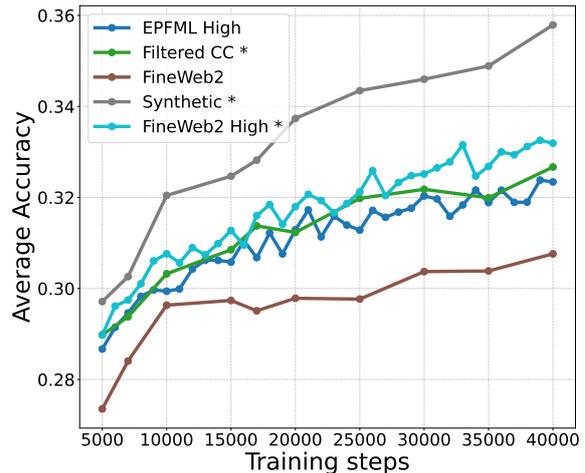


Figure 1: Average accuracies (MMMLU, ARC-Easy, HellaSwag; single- & five-shot) of 1B Llama-style models trained on ~ 84 billion tokens from different datasets. All subsets of GermanWeb – marked with an asterisk (*) – outperform FineWeb2. For comparison, we also include EPFML High. Here, 1,000 training steps equates to approximately 2.1 billion tokens. Individual benchmark results are provided in Appendix D.1.

We trained and evaluated our filtered Common Crawl (Filtered CC), synthetic, and our FineWeb2 High bucket, and compared these to random sampling German FineWeb2 and the top 10% of German FineWeb2 documents classified by Messmer et al. (2025) (EPFML High). The average accuracies across all benchmarks are shown in Figure 1 and the per-benchmark breakdowns are shown in Appendix D.1. We chose to exclude TruthfulQA from the 1B parameter model evaluations, as we found that it did not give a clear signal and suspect that this size of the model may be incapable of the task (*cf.* §3.3 for results of an 8B model).

We wish to emphasise that a large part of the synthetic subset’s average performance gain stems from its far better performance on the MMLU task. Its performance on ARC is still better than FineWeb2, but not clearly the best performer. And for the HellaSwag benchmark, while it again outperforms FineWeb2, high-quality organic data appear even better. This suggests that mixing differently-derived datasets may be appropriate to balance strengths and weaknesses of individual subsets.

3.3 8B HAT model

To evaluate our dataset on a different architecture and a larger scale, we pre-trained 8B hierarchical autoregressive transformer (HAT) models (Neite-meier et al., 2025a) (see Appendix C.2 for training

details). With this we leverage recent advances demonstrating that jointly training the tokeniser and model can be a promising alternative (Pagnoni et al., 2024), aligning with the philosophy of end-to-end learning (Sutton, 2019).

An additional benefit of using an end-to-end, tokeniser-free architecture like HAT is that it debiases comparisons between datasets—comparisons which are a central focus of this paper. Tokenisers have a significant impact on the efficacy of model training (Ali et al., 2024; Yang et al., 2024; Wang et al., 2024). Typically they are trained on large text corpora, which makes models using those tokenisers more effective on datasets that resemble the tokeniser’s training distribution. As a result, using a tokeniser-based model can introduce bias when comparing the effectiveness of different pre-training datasets, favouring those models that align more closely with the tokeniser’s original training data (Mayilvahanan et al., 2025). To avoid this bias, we use a HAT model, which does not rely on any pre-trained tokeniser and instead performs fixed, dataset-agnostic, byte-level string splitting.

Figure 2 compares the performance of these models pre-trained for different datasets and training scenarios. Differently to the 1B Llama-style models, these 8B HAT models operate on words, not tokens, resulting in 1,000 training steps equating to approximately three billion words in this set-up. We confirmed that our synthetic data on average outperforms randomly-sampled FineWeb2 data in Figure 2 (centre). We also confirmed our previous finding of our filtered CC data outperforming FineWeb2 at the 1B scale in Figure 2 (right).

When pre-training LLMs, web data is often mixed from different languages and with data from alternative, curated sources (Grattafiori et al., 2024), e.g., English and German data might be mixed with additional non-web data from books and encyclopedias. To test the performance of our single highest-performing subset, we compared our synthetic data to mix of 50% FineWeb2 and 50% high-quality curated datasets⁵. Figure 2 (left) shows our synthetic data outperformed this mixture on our tested benchmarks.

⁵In particular: the German National Library, European Union Parliament transcripts, European Patent Office applications, German Text Archive, Project Gutenberg books, JRC-Acquis, NewsAPI, Open Legal Data, Open Web Math, peS2o, PhilPapers, Wikibooks, Wikinews, Wikipedia, Wikisource, and Wikivoyage.

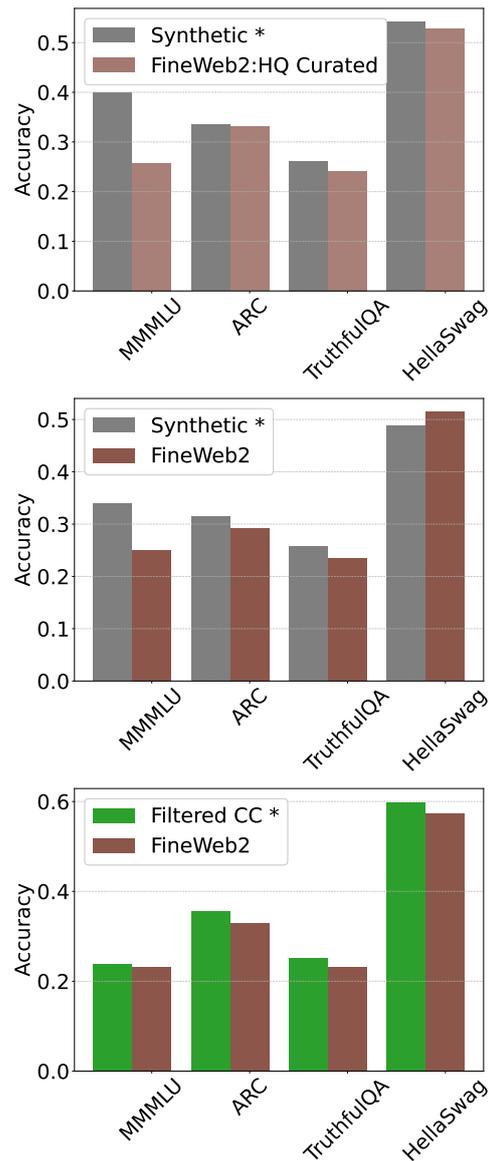


Figure 2: Accuracies (0-shot) of 8B HAT models trained on different datasets. On average, and for all but one individual benchmark comparison, GermanWeb subsets – marked with an asterisk (*) – outperform FineWeb2. **(Top)** We trained for 25,000 training steps on an English-language web-derived dataset, equating to ~ 75 billion English words. Afterwards the model was further trained for 20,000 steps (~ 60 billion words) with German language data, either random FineWeb2 data augmented with high quality datasets like Wikipedia (FineWeb2:HQ Curated), and listed further in Appendix C.3, or our generated synthetic data (Synthetic). Both datasets amount to 60 billion German words each. **(Middle)** We trained for 21,000 steps (~ 63 billion words) on either random FineWeb2 data or on our synthetic German data. **(Bottom)** The FineWeb2 training run was continued to 50,000 steps and is compared to a training of 50,000 steps on data from our Filtered CC pipeline. Both runs each equate to ~ 150 billion words.

4 Conclusion

This paper presents a comprehensive data curation pipeline for German-language LLM pre-training, resulting in the creation of our high-quality German dataset, GermanWeb composed of three subsets drawn from Common Crawl, FineWeb2, and synthetic data conditioned on organic web documents. We evaluate each subset in isolation and benchmark them against FineWeb2 training demonstrating the effectiveness of our data curation pipeline and synthetic data generation in enhancing pre-training datasets. We trained two different model architectures on GermanWeb and FineWeb2 on a range of German-language benchmarks. Our results show that all three subsets of our dataset individually outperform FineWeb2, even when the latter is combined with high-quality human-curated data sources. This highlights the potential of future work (i) iterating on existing data curation pipelines, (ii) improving synthetic data generation, and (iii) mixing organic and synthetic data for advancing German-language LLM pre-training.

We make GermanWeb publicly available to the research community to contribute to further advancements in this field. Our findings support the importance of data curation and synthetic data generation in LLM pre-training, and we believe that our work will have a positive impact on the development of German-language LLMs.

5 Limitations

We believe there are many improvements and further research directions which will be important in this area. For example, we expect high-quality translation of existing, high-performing English datasets – such as explored in Wang et al. (2025) – to be a helpful augmentation method. In our manual inspection of machine-translated texts from English to German, however, we found a high number of errors. We therefore caution against training on large amounts of machine-translated data without also rigorously assessing its quality and naturalness.

Relatedly, a more thorough analysis of synthetic data and its apparent quality is needed. Recent work has demonstrated that recursively training models exclusively on synthetic data leads to a phenomenon termed ‘model collapse’ (Shumailov et al., 2024), wherein model performance gradually deteriorates over successive generations of training and synthesis. However, mixing synthetic data

with organic data appears to avoid this issue (Gerstgrasser et al., 2024). Still, it remains unclear to what degree practitioners should opt for organic rather than synthetic data, or whether synthetic data is merely ‘gaming’ popular evaluation methods by providing data in a more relevant syntax than organic data.

References

- Amro Abbas, Kushal Tirumala, Dániel Simig, Surya Ganguli, and Ari S. Morcos. 2023. *Semdedup: Data-efficient learning at web-scale through semantic deduplication*. *Preprint*, arXiv:2303.09540.
- Mehdi Ali, Michael Fromm, Klaudia Thellmann, Richard Rutmann, Max Lübbering, Johannes Leveiling, Katrin Klug, Jan Ebert, Niclas Doll, Jasper Buschhoff, and 1 others. 2024. Tokenizer choice for llm training: Negligible or crucial? In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 3907–3924.
- Loubna Ben Allal, Anton Lozhkov, Elie Bakouch, Gabriel Martín Blázquez, Guilherme Penedo, Lewis Tunstall, Andrés Marafioti, Hynek Kydlíček, Agustín Piqueres Lajarín, Vaibhav Srivastav, Joshua Lochner, Caleb Fahlgren, Xuan-Son Nguyen, Clémentine Fourrier, Ben Burtenshaw, Hugo Larcher, Haojun Zhao, Cyril Zakka, Mathieu Morlon, and 3 others. 2025. *Smollm2: When smol goes big – data-centric training of a small language model*. *Preprint*, arXiv:2502.02737.
- Aleph Alpha. 2024. Pharia-1-llm-7b-control. <https://huggingface.co/Aleph-Alpha/Pharia-1-LLM-7B-control>. Accessed: 2025-04-17.
- Adrien Barbaresi. 2021. *Trafilatura: A web scraping library and command-line tool for text discovery and extraction*. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: System Demonstrations*, pages 122–131, Online. 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.
- Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*.

- A.Z. Broder. 1997. [On the resemblance and containment of documents](#). In *Proceedings. Compression and Complexity of SEQUENCES 1997 (Cat. No.97TB100171)*, pages 21–29.
- 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*.
- Harm De Vries. 2023. Go smol or go home, 2023. *URL* <https://www.harmdevries.com/post/model-size-vs-compute-overhead>.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. [BERT: Pre-training of deep bidirectional transformers for language understanding](#). In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.
- Peng Di, Jianguo Li, Hang Yu, Wei Jiang, Wenting Cai, Yang Cao, Chaoyu Chen, Dajun Chen, Hongwei Chen, Liang Chen, Gang Fan, Jie Gong, Zi Gong, Wen Hu, Tingting Guo, Zhichao Lei, Ting Li, Zheng Li, Ming Liang, and 19 others. 2024. [Codefuse-13b: A pretrained multi-lingual code large language model](#). In *Proceedings of the 46th International Conference on Software Engineering: Software Engineering in Practice, ICSE-SEIP '24*, page 418–429, New York, NY, USA. Association for Computing Machinery.
- Common Crawl Foundation. [Common crawl](#).
- Matthias Gerstgrasser, Rylan Schaeffer, Apratim Dey, Rafael Rafailov, Henry Sleight, John Hughes, Tomasz Korbak, Rajashree Agrawal, Dhruv Pai, Andrey Gromov, Daniel A. Roberts, Diyi Yang, David L. Donoho, and Sanmi Koyejo. 2024. [Is model collapse inevitable? breaking the curse of recursion by accumulating real and synthetic data](#). *Preprint*, arXiv:2404.01413.
- Aristides Gionis, Piotr Indyk, Rajeev Motwani, and 1 others. 1999. Similarity search in high dimensions via hashing. In *Vldb*, volume 99, pages 518–529.
- Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 542 others. 2024. [The llama 3 herd of models](#). *Preprint*, arXiv:2407.21783.
- Michael Hassid, Tal Remez, Jonas Gehring, Roy Schwartz, and Yossi Adi. 2024. [The larger the better? improved LLM code-generation via budget reallocation](#). In *First Conference on Language Modeling*.
- Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2020. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*.
- Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. [Measuring massive multitask language understanding](#). *Preprint*, arXiv:2009.03300.
- Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, and 3 others. 2022. [Training compute-optimal large language models](#). *Preprint*, arXiv:2203.15556.
- Chandra Irugalbandara, Ashish Mahendra, Roland Daynauth, Tharuka Kasthuri Arachchige, Jayanaka Dantanarayana, Krisztian Flautner, Lingjia Tang, Yiping Kang, and Jason Mars. 2024. [Scaling down to scale up: A cost-benefit analysis of replacing openai’s llm with open source slms in production](#). In *2024 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, pages 280–291.
- Joseph Jennings, Mostofa Patwary, Sandeep Subramanian, Shrimai Prabhume, Ayush Dattagupta, Vibhu Jawa, Jiwei Liu, Ryan Wolf, Sarah Yurick, and Varun Singh. [NeMo-Curator: a toolkit for data curation](#).
- Katherine Lee, Daphne Ippolito, Andrew Nystrom, Chiyuan Zhang, Douglas Eck, Chris Callison-Burch, and Nicholas Carlini. 2022. [Deduplicating training data makes language models better](#). *Preprint*, arXiv:2107.06499.
- 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-1m: In search of the next generation of training sets for language models](#). *Preprint*, arXiv:2406.11794.
- Stephanie Lin, Jacob Hilton, and Owain Evans. 2021. [Truthfulqa: Measuring how models mimic human falsehoods](#). *Preprint*, arXiv:2109.07958.
- Lin Long, Rui Wang, Ruixuan Xiao, Junbo Zhao, Xiao Ding, Gang Chen, and Haobo Wang. 2024. [On llms-driven synthetic data generation, curation, and evaluation: A survey](#). *Preprint*, arXiv:2406.15126.
- Pratyush Maini, Skyler Seto, He Bai, David Grangier, Yizhe Zhang, and Navdeep Jaitly. 2024. [Rephrasing the web: A recipe for compute and data-efficient language modeling](#). *Preprint*, arXiv:2401.16380.
- Max Marion, Ahmet Üstün, Luiza Pozzobon, Alex Wang, Marzieh Fadaee, and Sara Hooker. 2023. [When less is more: Investigating data pruning for pre-training llms at scale](#). *Preprint*, arXiv:2309.04564.

- Prasanna Mayilvahanan, Thaddäus Wiedemer, Sayak Mallick, Matthias Bethge, and Wieland Brendel. 2025. Lms on the line: Data determines loss-to-loss scaling laws. *arXiv preprint arXiv:2502.12120*.
- Bettina Messmer, Vinko Sabolčec, and Martin Jaggi. 2025. [Enhancing multilingual llm pretraining with model-based data selection](#). *Preprint*, arXiv:2502.10361.
- Joss Moorkens, Andy Way, and Séamus Lankford. 2024. *Automating Translation*. Routledge.
- Niklas Muennighoff, Alexander M. Rush, Boaz Barak, Teven Le Scao, Aleksandra Piktus, Nouamane Tazi, Sampo Pyysalo, Thomas Wolf, and Colin Raffel. 2023. [Scaling data-constrained language models](#). *Preprint*, arXiv:2305.16264.
- Pit Neitemeier, Björn Deiseroth, Constantin Eichenberg, and Lukas Balles. 2025a. Hierarchical autoregressive transformers for tokenizer-free language modelling. In *The Thirteenth International Conference on Learning Representations*.
- Pit Neitemeier, Björn Deiseroth, Constantin Eichenberg, and Lukas Balles. 2025b. [Hierarchical autoregressive transformers: Combining byte- and word-level processing for robust, adaptable language models](#). *Preprint*, arXiv:2501.10322.
- Binh-Nguyen Nguyen and Yang He. 2025. [Swift cross-dataset pruning: Enhancing fine-tuning efficiency in natural language understanding](#). *Preprint*, arXiv:2501.02432.
- Artidoro Pagnoni, Ram Pasunuru, Pedro Rodriguez, John Nguyen, Benjamin Muller, Margaret Li, Chunting Zhou, Lili Yu, Jason Weston, Luke Zettlemoyer, and 1 others. 2024. Byte latent transformer: Patches scale better than tokens. *arXiv preprint arXiv:2412.09871*.
- Guilherme Penedo, Hynek Kydlíček, Vinko Sabolčec, Bettina Messmer, Negar Foroutan, Amir Hossein Kargaran, Colin Raffel, Martin Jaggi, Leandro Von Werra, and Thomas Wolf. 2025. [Fineweb2: One pipeline to scale them all — adapting pre-training data processing to every language](#). In *Second Conference on Language Modeling*.
- Guilherme Penedo, Hynek Kydlíček, Loubna Ben alal, 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](#). In *Advances in Neural Information Processing Systems*, volume 37, pages 30811–30849. Curran Associates, Inc.
- Guilherme Penedo, Quentin Malartic, Daniel Hesslow, Ruxandra Cojocaru, Alessandro Cappelli, Hamza Alobeidli, Baptiste Pannier, Ebtesam Almazrouei, and Julien Launay. 2023. [The refinedweb dataset for falcon llm: Outperforming curated corpora with web data, and web data only](#). *Preprint*, arXiv:2306.01116.
- Björn Plüster. Germanbenchmark: Translating popular llm benchmarks to german. <https://github.com/bjoernpl/GermanBenchmark>. Accessed: 2025-04-17.
- 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. 2022. [Scaling language models: Methods, analysis & insights from training gopher](#). *Preprint*, arXiv:2112.11446.
- Siddharth Samsi, Dan Zhao, Joseph McDonald, Baolin Li, Adam Michaleas, Michael Jones, William Bergeron, Jeremy Kepner, Devesh Tiwari, and Vijay Gadeppally. 2023. [From words to watts: Benchmarking the energy costs of large language model inference](#). In *2023 IEEE High Performance Extreme Computing Conference (HPEC)*, pages 1–9.
- Ilia Shumailov, Zakhar Shumaylov, Yiren Zhao, Nicolas Papernot, Ross Anderson, and Yarin Gal. 2024. Ai models collapse when trained on recursively generated data. *Nature*, 631(8022):755–759.
- Shaden Smith, Mostofa Patwary, Brandon Norick, Patrick LeGresley, Samyam Rajbhandari, Jared Casper, Zhun Liu, Shrimai Prabhumoye, George Zerveas, Vijay Korthikanti, Elton Zhang, Rewon Child, Reza Yazdani Aminabadi, Julie Bernauer, Xia Song, Mohammad Shoeybi, Yuxiong He, Michael Houston, Saurabh Tiwary, and Bryan Catanzaro. 2022. [Using deepspeed and megatron to train megatron-turing nlg 530b, a large-scale generative language model](#). *Preprint*, arXiv:2201.11990.
- Ben Sorscher, Robert Geirhos, Shashank Shekhar, Surya Ganguli, and Ari S. Morcos. 2023. [Beyond neural scaling laws: beating power law scaling via data pruning](#). *Preprint*, arXiv:2206.14486.
- Dan Su, Kezhi Kong, Ying Lin, Joseph Jennings, Brandon Norick, Markus Kliegl, Mostofa Patwary, Mohammad Shoeybi, and Bryan Catanzaro. 2024a. [Nemotron-cc: Transforming common crawl into a refined long-horizon pretraining dataset](#). *Preprint*, arXiv:2412.02595.
- Hui Su, Zhi Tian, Xiaoyu Shen, and Xunliang Cai. 2024b. [Unraveling the mystery of scaling laws: Part i](#). *Preprint*, arXiv:2403.06563.
- Richard Sutton. 2019. The bitter lesson. *Incomplete Ideas (blog)*, 13(1):38.
- Dixuan Wang, Yanda Li, Junyuan Jiang, Zepeng Ding, Guochao Jiang, Jiaqing Liang, and Deqing Yang. 2024. Tokenization matters! degrading large language models through challenging their tokenization. *arXiv preprint arXiv:2405.17067*.

Jiayi Wang, Yao Lu, Maurice Weber, Max Ryabini, David Adelani, Yihong Chen, Raphael Tang, and Pontus Stenetorp. 2025. [Multilingual language model pretraining using machine-translated data](#). *Preprint*, arXiv:2502.13252.

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.

Jin Yang, Zhiqiang Wang, Yanbin Lin, and Zunduo Zhao. 2024. Problematic tokens: Tokenizer bias in large language models. In *2024 IEEE International Conference on Big Data (BigData)*, pages 6387–6393. IEEE.

Yixin Ye, Zhen Huang, Yang Xiao, Ethan Chern, Shijie Xia, and Pengfei Liu. 2025. [Limo: Less is more for reasoning](#). *Preprint*, arXiv:2502.03387.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830*.

A Prompt for LLM-as-a-judge in quality filtering

Educational filter – LLM-as-a-judge prompt (1/2)

SYSTEM: Deine Aufgabe ist es zu bewerten, wie gut die deutsche Sprachqualität eines deutschen Textauszugs ist.

Gebe deine Bewertung in folgendem JSON-Format:

```
{
  "content_criticism": str (Falls es Schwächen der Antwort im Bezug auf Inhalt gibt, zum Beispiel plötzlicher Themenwechsel, unklarer Sinn, oder fehlende Informationen, nenne und belege sie anhand von Beispielen.),
```

```
  "content_grade": number (Eine Zahl zwischen 1 und 5, wobei 5 die beste und 1 die schlechteste Bewertung ist:
```

```
    5: Sehr kohärent und informativ, sehr hohe Qualität.
```

```
    4: Kohärent und informativ.
```

```
    3: Kleine Schwächen im Inhalt, aber der Sinn kann erfasst werden.
```

```
    2: Begrenzte Relevanz oder Genauigkeit, der Sinn kann nicht ganz erfasst werden.
```

```
    1: Völlig inkonsistent oder unsinnig.
```

```
),
```

```
{
  "language_criticism": str (Falls es Schwächen der Antwort im Bezug auf Sprache gibt, zum Beispiel Mischung verschiedener Sprachen, Schimpfwörter, unübliche Wortwahl, nenne und belege sie anhand von Beispielen.),
```

```
  "language_grade": number (Eine Zahl zwischen 1 und 5, wobei 5 die beste und 1 die schlechteste Bewertung ist:
```

```
    5: Sehr förmliche, objektive deutsche Sprache.
```

```
    4: Wenige, vernachlässigbare Sprachfehler.
```

Educational filter – LLM-as-a-judge prompt (2/2) continued

3: Es gibt sprachliche Schwächen, aber der Sinn kann gut erfasst werden.

2: Stark umgangssprachlich/Sprachmischung.

1: Grobe Sprachmischung, Schimpfwörter oder schlechte Wortwahl, der Sinn kann nicht ganz erfasst werden.

),
"orthography_criticism": str
(Falls es Schwächen der Antwort im Bezug auf Grammatik und Rechtschreibung oder Tippfehler gibt, nenne und belege sie anhand von Beispielen. Dazu gehören Worte, die getrennt geschrieben wurden obwohl sie zusammengeschrieben werden sollten, falsche Verbbeugung und falsche Deklination, etc.),

"orthography_grade": number (Eine Zahl zwischen 1 und 5, wobei 5 die beste und 1 die schlechteste Bewertung ist:

5: Keine Fehler.

4: wenige unbedeutende Fehler.

3: einige unbedeutende Fehler.

2: ein paar grobe Fehler, aber die Antwort ist verständlich.

1: so viele Fehler, dass die Antwort unverständlich ist.

),

USER: Textauszug:

{document}

ASSISTANT: Bewertungs-JSON:

{

B Prompts for synthetic data generation

Rephrasing – data generation prompt

Geben Sie mir für den folgenden Absatz eine vielfältige Umformulierung in hochwertiger deutscher Sprache, im Stil von Wikipedia. Beginne deine Antwort mit 'Umformulierung:'.

Text:

{document}

Umformulierung:

Summarisation – data generation prompt

Ihre Aufgabe ist es, den bereitgestellten Text gemäß diesen Anweisungen zu lesen und umzuformulieren:

- Versuchen Sie, eine komprimierte, aber genaue und informative Version des Originaltextes zu erstellen, keine vereinfachte Zusammenfassung.
- Erfassen und bewahren Sie die entscheidenden Informationen, Schlüsselkonzepte, wichtigen Werte und sachlichen Details im Originaltext auf und machen Sie ihn gleichzeitig lesbarer und zugänglicher.
- Behalten Sie Fachbegriffe, Fachvokabular und komplexe Konzepte bei.
- Bewahren Sie Beispiele, Erklärungen zu Denkschritten und unterstützende Beweise auf, um die Tiefe und den Kontext des Textes zu erhalten.
- Fügen Sie nur Informationen hinzu, die im Originaltext vorhanden sind. Fügen Sie keine neuen oder unbegründeten Behauptungen hinzu.
- Schreiben Sie im Klartext

Beginne deine Antwort mit 'Umformulierte Version:'.

Text:

{document}

Umformulierte Version:

Rephrasing in Wikipedia style – data generation prompt

Ihre Aufgabe ist es, einen neuen Text zu Wissen aus dem bereitgestellten Text zu verfassen, indem Sie diesen Anweisungen folgen:

- Schreibe den Text als Passagen mit leicht verständlichen und qualitativ hochwertigen deutsche Sätzen um, wie sie in Lehrbüchern und Wikipedia stehen.
- Konzentrieren Sie sich auf inhaltliche Disziplinen wie Geisteswissenschaften, Sozialwissenschaften, Naturwissenschaften, Technik, Ingenieurwesen, Mathematik, Recht und Recht, Wirtschaft, Management, Kunst, Bildung, Agrarwissenschaften, Politik und Geschichte.
- Ignorieren Sie Inhalte, die keine nützlichen Fakten oder Kenntnisse enthalten.
- Bewahren Sie Beispiele, Erklärungen von Denkprozessen und unterstützende Beweise auf, um die Tiefe und den Kontext des Textes zu erhalten.
- Fügen Sie keine Details hinzu oder ändern Sie sie. Wiederholen Sie nur, was bereits im Text steht.
- Schreiben Sie im Klartext.
- Fügen Sie keine Titel, Untertitel, Notizen oder Kommentare hinzu.

Nacheinander wird der Nutzer nun Texte bereitstellen, verfolge dann die Anweisungen und schreibe den neuen Text.

Beginne deine Antwort mit 'Neuer Text:'.

Text:
{document}

Neuer Text:

Formulating questions – data generation prompt

Lesen Sie den bereitgestellten Text, und stellen Sie allgemeine Fragen. Befolgen Sie die folgenden Anweisungen sorgfältig:

1. Die Fragen sollen auch ohne den Text als Allgemeinwissen beantwortet werden können.
2. Wenn das nicht möglich ist, formuliere zunächst einen kurzen Kontext, oder antworte lediglich 'nicht möglich'.
3. Stellen Sie möglichst diverse Fragen zu unterschiedlichen sachlichen Informationen, wichtigem Wissen oder konkreten Details im Text.
4. Beantworten Sie die Fragen direkt und in kurzer prägnanter Sprache.
5. Stellen Sie Fragen in den folgenden verschiedenen Kategorien: Ja/Nein, Multiple-Choice mit mehreren Optionen zur Auswahl, Vergleichs- und Offener Fragen.
6. Stellen Sie 8 Fragen mit Antworten, zwei zu jeder der beschriebenen Kategorien.
7. Beginne mit 'Fragen-Antwort-Paare:'.

Text:
{document}

Fragen-Antwort-Paare:

Extracting lists – data generation prompt

Überprüfen Sie den Text, und extrahieren Sie die wichtigsten Informationen auf Deutsch. Befolgen Sie diese Anweisungen:

- Lesen Sie den obigen Text sorgfältig durch und erstellen Sie eine prägnante und organisierte Liste mit sachlichen Informationen, konkreten Details, Schlüsselkonzepten sowie wichtigen Zahlen und Statistiken, die aus dem Text entnommen sind.
- Stellen Sie sicher, dass jeder Punkt klar und spezifisch ist und durch den Originaltext gestützt wird.
- Stellen Sie sicher, dass der extrahierte Text informationsdicht und leichter zu erlernen ist.
- Fügen Sie keine Titel oder Überschriften hinzu. Beginne deine Antwort mit 'Liste:'.

Text:

{document}

Liste:

C Experiments

C.1 1B Llama-like model training details

We trained a 1 billion parameter transformer model with 16 layers, 2,048 hidden dimensions, 32 attention heads, and 8 key-value heads. The model uses a context length of 4096 tokens and employs modern architecture choices: rotary positional embeddings (RoPE) with a base of 500,000, SwiGLU activation functions with an MLP expansion factor of 4, and RMSNorm for layer normalisation.

The model was trained for 40,000 iterations with a global batch size of 512. We used the AdamW optimiser with $\beta_1 = 0.9$, $\beta_2 = 0.95$, $\epsilon = 1e - 8$, and gradient clipping at 1.0. The learning rate followed a cosine decay schedule with an initial rate of $3e-4$, 400 warmup steps, 4,000 cooldown steps, and minimum learning rate of 0.0. We implemented ZeRO optimisation for distributed training efficiency. The model uses BFloat16 precision and was trained using a UTF-8 tokeniser with a vocabulary size of 131,072 tokens.

We used the Pharia-1 tokeniser that was trained with the Unigram algorithm using the Sentence-

Piece library (Alpha, 2024).

Parameter	Value
β_1	0.9
β_2	0.95
ϵ	1e-8
N_{Warmup}	1%
N_{Cooldown}	10%
learning rate	3e-4

Table 5: Parameters of the AdamW optimiser for the 1B-Llama-model.

Hyperparameter	Value
N_{vocab}	131,072
n_{Layers}	16
n_{heads}	32
d_{model}	2,048
d_{MLP}	8,192
Batch Size	512
Sequence Length	4,096

Table 6: Hyperparameters for the 1B-Llama-model.

C.2 8B HAT model training details

For the second part of our experiments, we trained 8B HAT models that consist of 3 consecutive transformers. In contrast to classical tokeniser-based models, HAT splits words by a rule-based splitter. Words are processed byte-wise by the smaller outer transformers, while the bigger backbone acts as the next-word predictor on a single embedding per word. The first and third models act as encoder and decoder, that transform bytes to word embeddings and vice versa. These models are connected by linear projection layers.

Words are split by special characters and punctuation, and converted into UTF-8 byte sequences.

All three models are trained simultaneously end-to-end, with hyperparameters similar to those in §C.2 and shown in the following tables.

Parameter	Value
β_1	0.9
β_2	0.95
ϵ	1e-8
N_{Warmup}	1%
N_{Cooldown}	10%
learning rate	1e-4

Table 7: Shared parameters of the AdamW optimiser for all transformer modules assembling the 8B-HAT-model.

Hyperparameter	Value
N_{vocab}	256
$n_{\text{Layers}}^{\text{Encoder}}$	6
$n_{\text{Layers}}^{\text{Decoder}}$	4
n_{heads}	6
d_{model}	768
d_{MLP}	2,047
Batch Size	1,024
Sequence Length	<i>variable</i>

Table 8: Hyperparameters for the Encoder/ Decoder Models of 8B-HAT. Almost all parameters are the same for the Encoder and Decoder. Only the number of layers is increased to 6 for the Encoder. As we fixate the number of words per training steps in the Backbone, the Encoder and Decoder will see a variable sequence length per training batch.

C.3 HQ datasets list

Books & literature (*e.g.*, Gutenberg); legal & government (*e.g.*, European Patent Office (EPO) and EU Parliament); news & web text (*e.g.*, Wikinews); encyclopedic & reference (*e.g.*, Wikipedia); scientific & technical (*e.g.*, peS2o); and parallel & multilingual (*e.g.*, ParaCrawl).

D Extended results

D.1 1B Llama-style model individual benchmarks results

GermanWeb outperforms FineWeb2 across all benchmarks. Interestingly, we find that different subsets have different strengths: our synthetic data performed particularly well on MMMLU, whereas

Hyperparameter	Value
N_{vocab}	-
n_{Layers}	32
n_{heads}	32
d_{model}	4,096
d_{MLP}	11,008
Batch Size	1,024
Sequence Length	3,500

Table 9: Hyperparameters for the Backbone Model of 8B-HAT.

our filtered CC and high-quality classified data performed best for HellaSwag and ARC.

D.2 Compute Resources

The training of our 1B-model for 40k steps, which in total amounts to roughly 8.4×10^{10} tokens, was performed for 9 datasets: Our 5 FineWeb2 quality buckets, random Fineweb, epfml-FineWeb2, Filtered CC, and our synthetic data. Computation was done with current standards in bfloat16 format. The total compute budget was 12 hours on 64 H100 GPUs per ablation.

The 8B HAT-model was trained for 3 different experiments with 5 runs in total. First, for 50k steps, which in total amount to roughly 1.5×10^{11} words for Filtered CC and FineWeb2. Secondly, for 21k steps, around 6.3×10^{10} words for synthetic data. The 21k snapshot from the FineWeb2 training was reused for this comparison. Lastly for 25k on English and 20k on German data, equating to roughly 1.35×10^{11} words in total. Note that each word is one token. The total compute budget was 4 days on 128 H100 GPUs per ablation.

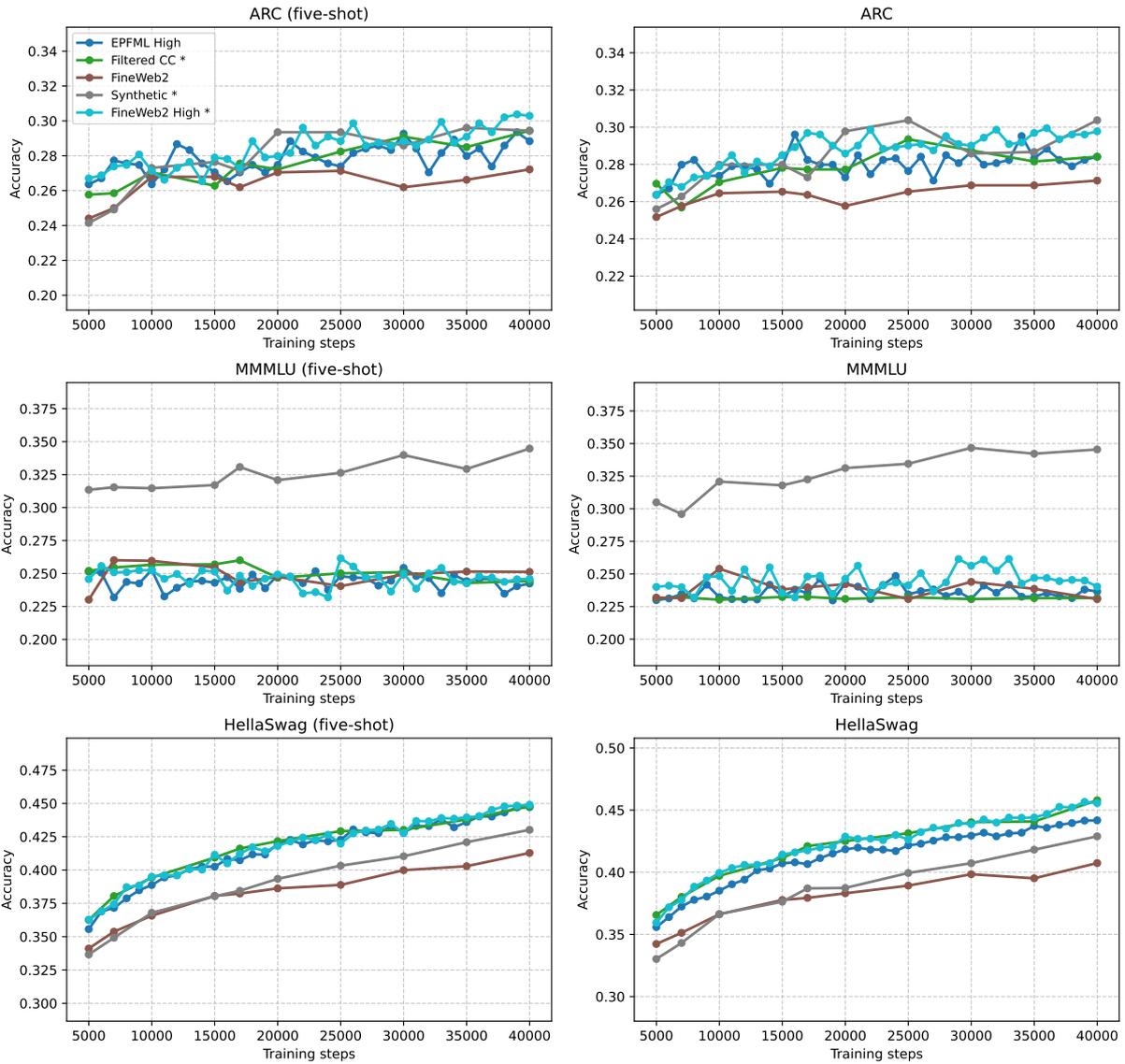


Figure 3: Individual benchmark results for 1B Llama-style models trained on ~ 84 billion tokens from different datasets. Here, 1,000 training steps equates to approximately 2.1 billion tokens.