TINY_SCHILLER: A Drop-In German Drama Corpus for Small Language Models

Mark Schutera
Duale Hochschule Baden-Württemberg Ravensburg
schutera@dhbw-ravensburg.de

Abstract

TINY_SCHILLER closes the small-language-model prototyping, fine-tuning, education, and research gap for German literary text, providing a single-file, drop-in counterpart to Karpathy's TINY_SHAKESPEARE. The available German literary corpora are larger and richer, but require parser engineering before a single line of training or fine-tuning code can run. TINY_SCHILLER is a 2.07-megabyte single file of eleven public-domain Schiller dramas, sourced from DraCor's GerDraCor export (CC0) and processed by deterministic parser engineering. Character-level, GPT-2 byte-pair encoding, and `cl100k_base` tokenization splits, an instruction-formatted dialogue-completion split, and 89 per-character persona splits load from a single HuggingFace call. A small language model literally reaches German literary text in one line of code.

1 Introduction

The German tiny-corpus gap. Small literary corpora are a fixture of small-scale language modelling. TINY_SHAKESPEARE (Karpathy, 2015), popularised through NANOGPT (Karpathy, 2022), lets a practitioner train a working transformer from scratch in minutes on a single laptop, and serves equally well as a fine-tuning target on a stylistically distinctive register. The BabyLM Challenge (Warstadt et al., 2023) has since established small fixed corpora as a first-class unit of LM research. The artifact behind TINY_SHAKESPEARE's reach is a single 1.1 MB file. The pipeline assumed by most educational and research material reads it directly.

No comparable single-file, cleaned, German counterpart exists. Practitioners who want to prototype, fine-tune, or use a non-English literary register for education and research currently fall back either on multi-gigabyte web crawls or on ad-hoc text downloads from sources such as Projekt Gutenberg-DE (Cultural Assets GmbH, 2026), without provenance, without documented cleaning, and without reproducible preprocessing. Larger and richer German corpora exist: The Deutsches Textarchiv (Berlin-Brandenburgische Akademie der Wissenschaften, 2007) and DraCor (Fischer et al., 2019; Börner et al., 2023) are deeper resources by every metric except one.

The Metric is Friction. If the German literary corpora are deeper, better-annotated, and longer-established than TINY_SHAKESPEARE, why has no German equivalent of TINY_SHAKESPEARE-scale small-LM emerged? Not because the data is missing. Because reaching the data costs an attention budget that small-scale prototyping, education, and research rather would not pay. For German, there is no single cleaned file that drops into a reproducible small-LM pipeline without parser engineering. The available German corpus resources require non-trivial preliminaries before a single token reaches a model: TEI/XML parsers, namespace handling, edition normalisation, speaker-tag reconciliation, encoding fixes for legacy umlauts and quote glyphs, and, in DraCor's case, a container stack to reproduce the corpus build at all (Börner et al., 2023). The work is not technically difficult. However, it is *upstream* of the modelling task and competes with it for a finite attention budget. In an educational setting it consumes the first session before any model has seen any text. In a prototyping or research setting it dominates the first day before any tokenizer has been compared. This paper refers to this aggregate of small, individually tractable preprocessing steps as *single-file friction*: The gap between "a corpus exists in some form" and "a corpus is ready for the downstream small-LM".

TINY_SHAKESPEARE's reach in English-language education and research does not come from its size. It comes from having no friction.

TINY_SCHILLER is built to remove the friction for German, which means contributing four small things together:

- **A cleaned, single-file release.** A 2.07 MB UTF-8 file produced by a deterministic pipeline, drop-in for small-LM trainers from scratch (including NANOGPT) and for standard fine-tuning pipelines (§3).
- **Tokenization splits matching standard small-LM workflows.** Character-level, GPT-2 BPE, and `cl100k_base` token streams are precomputed (§4).
- **Fine-tuning Parquets.** Whole-work, instruction-formatted dialogue-completion (`instruct.parquet`), and 89 per-character persona parquets are released for supervised fine-tuning (§5).
- **A reference fine-tune.** End-to-end usability on a single consumer GPU, documented with concrete numbers rather than asserted (§6).
- **An agent-ready data card.** A concise Markdown summary (`DATA_CARD.md`), structured as a data statement (Bender and Friedman, 2018), that documents provenance and intended use and enables quick, robust corpus ingestion by agentic tooling.[1]

Beyond friction removal, the BPE and `cl100k_base` splits double as a testbed for a known second-order effect: English-trained tokenizers encode archaic German poorly, a gap documented at scale (Rust et al., 2021; Petrov et al., 2023; Ahia et al., 2023). This paper reports the corpus-level fertility number for this specific testbed in §4.

2 On TINY_SCHILLER

TINY_SCHILLER comprises eleven dramatic works by Friedrich Schiller (Table 1), all in the public domain and sourced from DraCor's GerDraCor plaintext export (CC0).[2] The texts span Schiller's dramatic career from *Die Räuber* (1781) to *Wilhelm Tell* (1804) and include the canonical mature dramas. Poetry, prose fiction, historical writings, and aesthetic essays are excluded to keep the corpus stylistically homogeneous and dialog-dense.

[1] https://github.com/schutera/tiny_schiller.
[2] https://dracor.org/api/v1/corpora/ger/

Work	Personas
Die Räuber	20
Die Verschwörung des Fiesco zu Genua	25
Kabale und Liebe	10
Don Carlos, Infant von Spanien	34
Wallensteins Lager	15
Die Piccolomini	23
Wallensteins Tod	24
Maria Stuart	23
Die Jungfrau von Orleans	39
Die Braut von Messina (test split)	11
Wilhelm Tell (test split)	51

Table 1: Works included in TINY_SCHILLER and the number of distinct speaker personas (with at least one turn) present in each work.

Schiller is a deliberate *single-author* choice. The eleven dramas are stylistically homogeneous, populated by a small set of vivid characters whose voices remain consistent across acts, and exercise the orthographic edges of German (umlauts, the eszett, French loanword diacritics, and « angle quotes ») that modern news German rarely touches. Goethe and Lessing are reasonable alternative choices; this point is revisited in §7.

3 Making the File Drop-In

`tiny_schiller.parsed.txt`, the single-file artifact: 2,019,857 characters in 2.07 MB of UTF-8 (no BOM), 1.88× TINY_SHAKESPEARE in bytes but fewer tokens under BPE (§4). "No BOM" means the file does not begin with a UTF-8 byte-order mark, which avoids spurious leading characters in downstream tooling. All other released artifacts are deterministic derivatives.

The file is produced by `scripts/parse.py` from the concatenated raw downloads, performing (a) mechanical normalisation of upstream typesetting artefacts and (b) unification of three speaker-tag styles into one canonical form. Each step

TINY_SCHILLER Statistic	Value
FILE characters	2,019,857
FILE bytes (UTF-8)	2,067,041
FILE encoding	UTF-8 (no BOM)
FILE line endings	LF
CHARSET unique codepoints	88
CHARSET « / »	101 / 101
DRAMA works	11
DRAMA speaker-tag format	`SPEAKER:\n`

Table 2: `tiny_schiller.parsed.txt` summary statistics: File, character-set, and dramatic-structure properties.

is small in isolation; together they eliminate the parser-engineering burden described in §1.

Mechanical normalisation. Convert CRLF to LF; collapse ≥3 blank lines to two; strip the narrow no-break space (NNBSP, U+202F) found in some upstream editions; and normalise en-dashes. These reversible byte-level edits match the conventions assumed by tokenizer training scripts and small-LM preparation utilities.

Speaker-tag unification. Upstream editions mix three styles (inline SPEAKER. TEXT, standalone SPEAKER., and standalone SPEAKER:); all are rewritten into `SPEAKER:\ntext`, the TINY_SHAKESPEARE/NANOGPT convention. The regular speaker boundary feeds the per-character splits (§5) and matches common assumptions in computational drama analysis (Moretti, 2011; Fischer et al., 2019).

4 Three Tokenization Splits

This release provides three precomputed token streams for standard small-LM workflows: A character baseline and two widely used subword encodings. Each split follows a deterministic 90/10 train/validation partition and writes `train.bin` / `val.bin` in the NANOGPT convention. Table 3 reports vocabulary size and tokenization efficiency (characters per token) on `tiny_schiller.parsed.txt`.

Default and takeaway. Unless stated otherwise, this paper uses `schiller_bpe` for token-based reporting in this paper, since GPT-2 BPE is a widely used and stable baseline in small-LM tooling. The three splits are a character-level baseline (`schiller_char`, vocabulary 88), GPT-2 BPE (Radford et al., 2019) via `tiktoken` (Sennrich et al., 2016; OpenAI, 2022) (`schiller_bpe`), and `cl100k_base` (`schiller_cl100k`). The `cl100k_base` split is included to quantify context-budget effects: On this corpus it yields 3.14 chars/-tok vs. 2.36 for GPT-2 BPE, reflecting the known cross-lingual inefficiency of English-trained tokenizers on German documented at scale elsewhere (Rust et al., 2021; Petrov et al., 2023; Ahia et al., 2023).

5 Fine-Tuning Parquets

Loading any of the data splits from the HuggingFace Hub takes a single line:

```
from datasets import load_dataset

ds = load_dataset("mrkschtr/tiny_schiller",
    split="train")
```

Three parquet splits. Where §4 provides *token streams* for training from scratch, this section provides *example datasets* for supervised fine-tuning. The release ships three parquet splits on the HuggingFace Hub,[3] all derived deterministically from `tiny_schiller.parsed.txt` by `scripts/build_instruct.py`: A whole-work split, an instruction-formatted dialogue-completion split (`instruct.parquet`), and 89 per-character persona splits.

Schemas. The whole-work split exposes a single TEXT field per row and holds out *Wilhelm Tell* and *Die Braut von Messina* as the test partition. `instruct.parquet` exposes PROMPT and COMPLETION fields, with WORK and CHARACTER as filtering metadata. The 89 per-character files share that schema and contain only the rows whose target speaker matches the named character.

Construction. The instruct and persona rows are produced by a single sliding window over the canonical speaker-tagged file: The prompt fills one of six non-persona dialogue-continuation templates with three preceding speaker turns from one work, and the completion is the next speaker's turn, formatted as `NAME:\ntext`. The per-character files use the same window with four character-specific persona templates.

From split to trainer. The instruct and persona splits load through the same call by pointing the loader at the appropriate parquet file under `data/`. The two schemas reflect different training framings: The whole-work split is shaped for next-token training, the instruct and persona splits for prompt/completion SFT. A one-line map that concatenates the prompt and completion fields into a single text field unifies them before tokenization.

[3] https://huggingface.co/datasets/mrkschtr/tiny_schiller.

Pipeline	Vocab	chars/tok
`schiller_char`	88	1.00
`schiller_bpe`	50,257	2.36
`schiller_cl100k`	≈100k	3.14

Table 3: `tiny_schiller.parsed.txt` tokenizer comparison.

6 A Reference Fine-Tune

The reference fine-tune is the proof that the one-screen pitch holds: Data drop-in, supervised fine-tuning, and a working stylistic checkpoint, all on a single consumer GPU. The full training script fits on a screen:

```
from datasets import load_dataset
from transformers import AutoModelForCausalLM,
    TrainingArguments
from trl import SFTTrainer

ds = load_dataset("mrkschtr/tiny_schiller",
    data_files="data/instruct.parquet", split="
    train")
model = AutoModelForCausalLM.from_pretrained("
    Qwen/Qwen2.5-0.5B-Instruct")
args = TrainingArguments(output_dir="out",
    per_device_train_batch_size=4,
    num_train_epochs=2)
SFTTrainer(model, ds, args).train()
```

The reference run is a two-stage SFT. Stage 1 trains on the full `instruct.parquet` to teach the base model the dialogue-continuation register; stage 2 specialises a copy of that checkpoint on a single per-character file (`char_MOOR.parquet`) to demonstrate persona adaptation. The two stages exercise the two intended fine-tuning workflows from §5 (corpus-wide stylistic adaptation and per-character specialisation) in one pipeline, and reports concrete numbers rather than asserting feasibility.

Persona adaptation completes in three minutes (Table 4), cheap enough to iterate on interactively; the full two-stage run still fits inside a single session on accessible hardware.

Parameter	Value
Base model	`Qwen/Qwen2.5-0.5B-Instruct`
Stage 1 dataset	`instruct.parquet`, 7,454 train / 153 eval (2% holdout)
Stage 2 dataset	`char_MOOR.parquet`, 127 train / 3 eval
Epochs	1. Stage: 2, 2. Stage: 2
Early stopping	eval_loss, patience 2, eval every 100 steps (not triggered)
Batch size	4
Context length	512 tokens
Learning rate	2×10^{-4} (Stage 1), 1×10^{-4} (Stage 2)
Weight decay	0.05 (Stage 2 only)
Hardware	NVIDIA RTX 3060 (12 GB VRAM, bf16 precision)
Training time	3.6 hours total (Stage 1: 3 h 34 min, Stage 2: 3 min)
Final eval loss	0.182 (Stage 1) / 0.044 (Stage 2)
Final token accuracy	0.965 (Stage 1) / 0.989 (Stage 2)

Table 4: Reference fine-tune: Parameters, hardware, and final-checkpoint metrics.

7 Where This Helps and Where It Doesn't

Given the artifacts of §5 and the reference run of §6, TINY_SCHILLER is intended for drop-in use in small-LM prototyping, producing coherent in-register continuations from a small open base model. The most direct use cases include:

- **Rapid prototyping** of architectures and tokenization choices in low-compute settings.
- **Stylistic fine-tuning** of pretrained German or multilingual models on a homogeneous literary register.
- **Per-character persona fine-tuning** using the prebuilt per-character splits.
- **Education and research**, including small-scale language modelling studies on a non-English literary register.

The contribution is operational rather than evaluative: No claim is made about generation quality. As an illustrative random sample, the Karl-Moor persona fine-tune (0.5B model, §6) continues the prompt *Freiheit?* as follows:

> **Moor**
> *aufspringend. Frein! Bist du wahnsinnig? Willst du dich von deinen Bubenjahren hofmeistern? Komm näher – zögerst du wieder vom Pferd, wo die Ehre ist?*
>
> **Schweizer:**
> *Gebeut, Hauptmann! was sollen wir tun?*
>
> **Moor:**
> *Steh auf, Schweizer! und rührt sich der Schaum. Wir müssen durch diese Nacht noch zurückkommen.*

The corpus is too small to pretrain a competitive general-purpose language model and does not constitute a benchmark in the sense of a held-out evaluation suite. Speaker-tag detection is regex-based rather than grammar-based, so while it normalises the most common plain-text edition styles, it remains a heuristic; edge-case typography, such as stage directions formatted as speaker turns, survives into the released file and should be expected.

Beyond Schiller. The same toolchain applies to other public-domain drama with explicit speaker tags, such as Goethe, Lessing, and Kleist, and to German translations of international authors, enabling future TINY_⟨AUTHOR⟩ releases for cross-author comparison at the same scale. Minor adaptations extend it to non-dramatic text and to additional languages.

References

Orevaoghene Ahia, Sachin Kumar, Hila Gonen, Jungo Kasai, David R. Mortensen, Noah A. Smith, and Yulia Tsvetkov. 2023. Do all languages cost the same? Tokenization in the era of commercial language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9904–9923. Association for Computational Linguistics.

Emily M. Bender and Batya Friedman. 2018. Data statements for natural language processing: Toward mitigating system bias and enabling better science. *Transactions of the Association for Computational Linguistics (TACL)*, 6:587–604.

Berlin-Brandenburgische Akademie der Wissenschaften. 2007. Deutsches textarchiv (DTA). `https://www.deutschestextarchiv.de/`. Ongoing project.

Ingo Börner, Peer Trilcke, Carsten Milling, Frank Fischer, and Henny Sluyter-Gäthje. 2023. Dockerizing DraCor: A container-based approach to reproducibility in computational literary studies. In *Proceedings of the Digital Humanities Conference (DH2023)*, Graz, Austria.

Cultural Assets GmbH. 2026. Projekt gutenberg-DE. `https://www.projekt-gutenberg.org/`. Formerly operated by Hille and Partner.

Frank Fischer, Ingo Börner, Mathias Göbel, Angelika Hechtl, Christopher Kittel, Carsten Milling, and Peer Trilcke. 2019. Programmable corpora: Introducing DraCor, an infrastructure for the research on european drama. In *Proceedings of the Digital Humanities Conference (DH2019)*, Utrecht, Netherlands.

Andrej Karpathy. 2015. The unreasonable effectiveness of recurrent neural networks. `https://karpathy.github.io/2015/05/21/rnn-effectiveness/`. Blog post; introduces `tiny_shakespeare`.

Andrej Karpathy. 2022. nanoGPT. `https://github.com/karpathy/nanoGPT`.

Franco Moretti. 2011. Network theory, plot analysis. Pamphlet 2, Stanford Literary Lab.

OpenAI. 2022. tiktoken: A fast BPE tokeniser for use with OpenAI's models. `https://github.com/openai/tiktoken`.

Aleksandar Petrov, Emanuele La Malfa, Philip H. S. Torr, and Adel Bibi. 2023. Language model tokenizers introduce unfairness between languages. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. Technical report, OpenAI.

Phillip Rust, Jonas Pfeiffer, Ivan Vulić, Sebastian Ruder, and Iryna Gurevych. 2021. How good is your tokenizer? On the monolingual performance of multilingual language models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (ACL-IJCNLP)*, pages 3118–3135. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Alex Warstadt, Aaron Mueller, Leshem Choshen, Ethan Wilcox, Chengxu Zhuang, Juan Ciro, Rafael Mosquera, Bhargavi Paranjabe, Adina Williams, Tal Linzen, and Ryan Cotterell. 2023. Findings of the BabyLM challenge: Sample-efficient pretraining on developmentally plausible corpora. In *Proceedings of the BabyLM Challenge at CoNLL 2023*, pages 1–34, Singapore. Association for Computational Linguistics.