A Single-GPU LM for Linux Documentation

Nikolaos Sopiadis

1. Introduction

I present a compact, decoder-only Transformer language model tailored to Linux documentation, trained end-to-end on a single RTX 3060 GPU. This project's components include

- a streaming data pipeline that reads large raw text in 1 MiB chunks and applies dynamic 25% MLM masking
- a 6-layer, 13 M-parameter causal Transformer with sinusoidal position encodings
- a 6-layer, 25 M-parameter causal Transformer with sinusoidal position encodings
- a multi-epoch training regimen using gradient accumulation, cosine-decay learning-rate schedule, and mixed-precision.
- an interactive console where you can test the models by providing them with prompts.

2. Related Work

Cramming [2] showed that a 110 M-parameter BERT-style model can be trained in one day on a single RTX 2080 Ti using a one-pass over billions of tokens, disabling dropout and using a linear LR schedule. Unlike these, my focus is on a *small-scale, domain-specific* LM trained over *multiple epochs* on tens of millions of tokens with a decoder-only causal architecture, while also utilizing the optimizations discussed on the cramming paper.

3. Formatting your Response

4. Method

4.1. Dataset

I gathered Linux related text (170 MB train, 17 MB eval), striped markup (html elements), and stored as '.txt'. They were further processed to strip all non english characters and only files larger than 1 KiB where kept. The dataset comes from the following sources:

- Archwiki
- Man-pages
- Linux Kernel Documentation
- Select few Wikipedia articles.

4.2. Data pipeline

`StreamingMLMDataset` reads each file in 1 MiB chunks, tokenizes with HuggingFace's `BertTokenizerFast`, buffers token IDs, and emits non-overlapping 128-token sequences. Dynamic masking (rate=25% as per the cramming paper [2]) follows Devlin et al.'s 80/10/10 rule [1].

4.3. Model Architecture

The `TransformerLM` stacks six identical blocks. Each block uses *pre-norm* residuals:

$$
\begin{aligned}
\tilde{X} &= \mathrm{LN}(X), \\
Y &= X + \mathrm{Dropout}\big(\mathrm{MHA}(\tilde{X})\big), \\
\tilde{Y} &= \mathrm{LN}(Y), \\
Z &= Y + \mathrm{FFN}(\tilde{Y}),
\end{aligned}
$$

where MHA has 4 heads ($256/4 = 64$ for 13M and $512/4 = 128$ for 25M each), and FFN is two linear layers ($256 \to 1024 \to 256$ for 13M and $512 \to 2048 \to 512$) with GELU and dropout (0.1). Positional encodings are fixed sinusoids [3]. Token embeddings and the final linear head are weight-tied, bringing the total to ≈ 13 and ≈ 25 M parameters respectively. Another optimization used here, is omitting the Q/K/V biases, as they affect minimally model efficiency and significantly increase training performance. Finally, both models have a gradient cutoff threshold. For the 13 M-param model, the threshold remains constantly at 0.5[2] and for the 25 M-param model, the threshold is decreased linearly from 1.5 to 0.5.

4.4. Training Setup

The model is trained for 10 epochs (for experimentation I stopped the training later than supposed to) with batch accumulation ramped linearly from 21 to 85 micro-batches for the 13M model -as per cramming [2]- and from 10 to 40 for the 25M model micro-batches. Each micro-batch contains 96 tokens giving us effectively $\approx 2K \to 8K$ and $\approx 1K \to 4K$ tokens respectively. Optimizer: AdamW ($\beta_1 = 0.9, \beta_2 = 0.98, \epsilon = 1e-12, wd = 0.01$) in both cases, as per cramming [2]. LR schedule: cosine decay with 10% warmup across all updates. Other optimizations include the use of mixed precision floating point numbers and the JIT compilation of the network to accelerate training. The metrics used to evaluate performance are: loss, PPL, masked-token accuracy, top-5 accuracy.

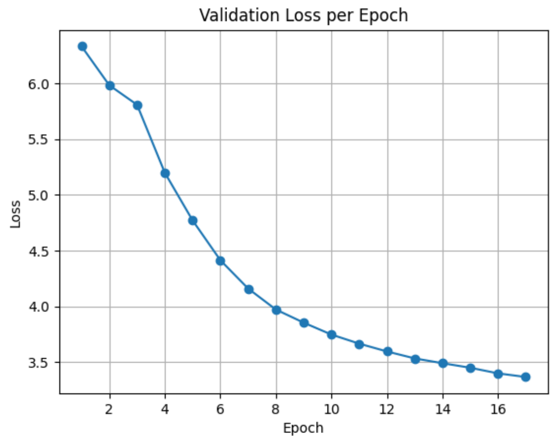


Figure 1. Validation loss per epoch (13 M).

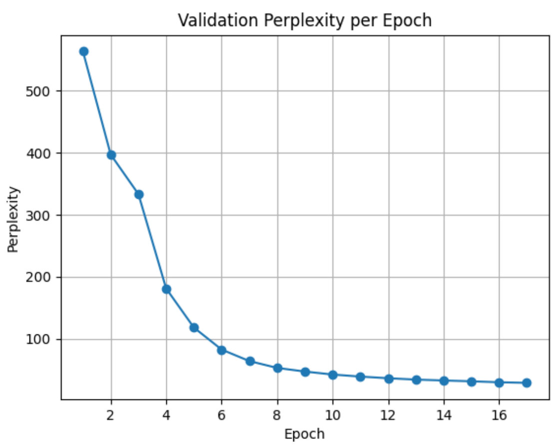


Figure 2. Validation perplexity per epoch (13 M).

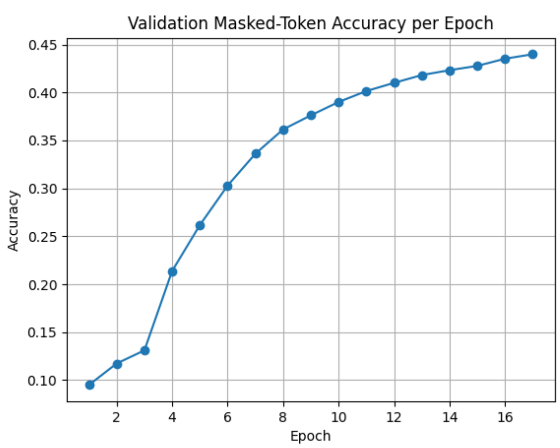


Figure 3. Masked-token accuracy per epoch (13 M).

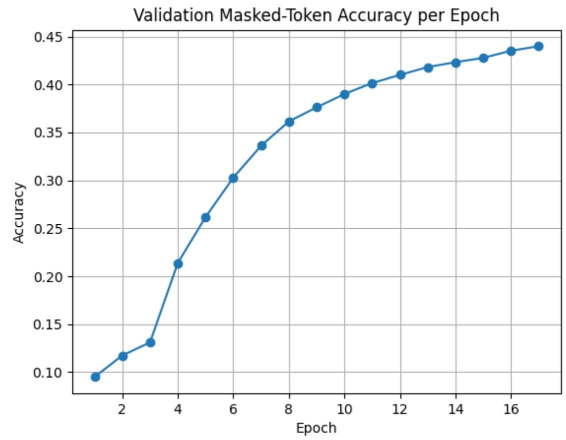


Figure 4. Top-5 accuracy per epoch (13 M).

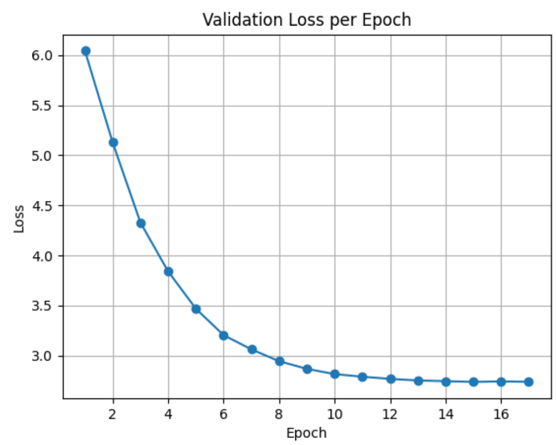


Figure 5. Validation loss per epoch (25 M).

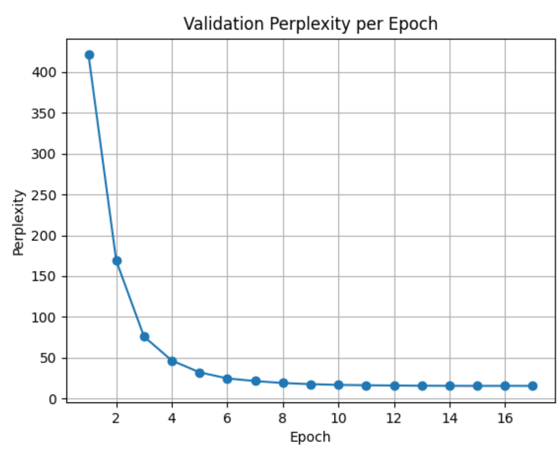


Figure 6. Validation perplexity per epoch (25 M).

5. Experimental Results

5.1. Quantitative Metrics

5.2. Qualitative Examples

5.3. Evaluation

Neither models' outputs are valid. The first was trained first and since it did not perform as expected, the second

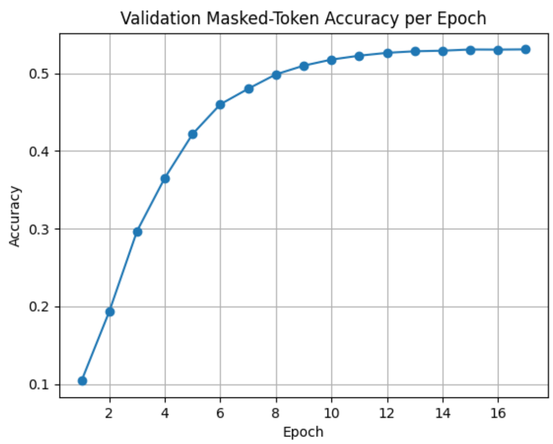


Figure 7. Masked-token accuracy per epoch (25 M).

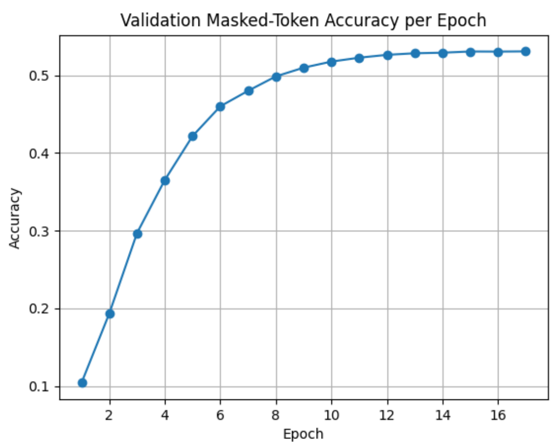
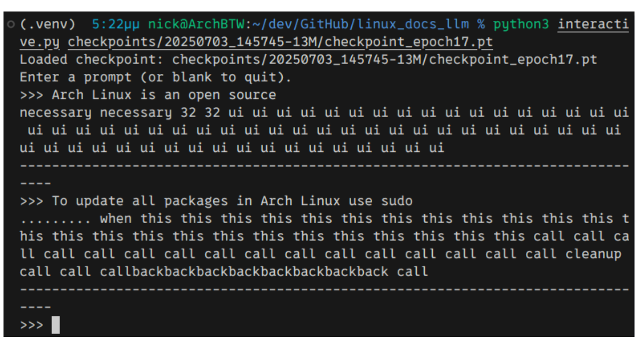


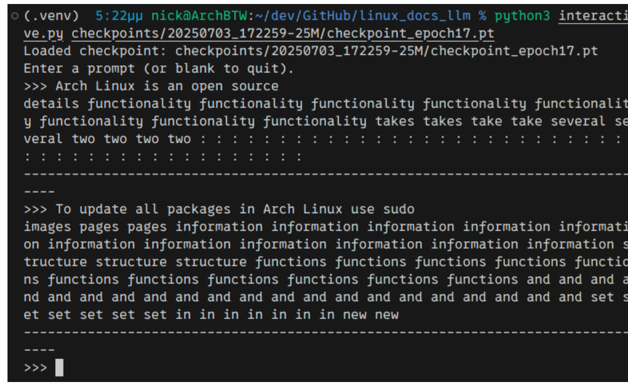
Figure 8. Top-5 accuracy per epoch (25 M).



```
○ (.venv)  5:22μμ nick@ArchBTW:~/dev/GitHub/linux_docs_llm % python3 interacti
ve.py checkpoints/20250703_145745-13M/checkpoint_epoch17.pt
Loaded checkpoint: checkpoints/20250703_145745-13M/checkpoint_epoch17.pt
Enter a prompt (or blank to quit).
>>> Arch Linux is an open source
necessary necessary 32 32 ui ui ui ui ui ui ui ui ui ui ui ui ui ui ui ui ui
 ui ui ui ui ui ui ui ui ui ui ui ui ui ui ui ui ui ui ui ui ui ui ui ui ui
ui ui ui ui ui ui ui ui ui ui ui ui ui ui ui ui ui ui
--------------------------------------------------------------------------------
----
>>> To update all packages in Arch Linux use sudo
......... when this this this this this this this this this this this this t
his this this this this this this this this this this this this call call ca
ll call call call call call call call call call call call call call cleanup
call call callbackbackbackbackbackbackback call
--------------------------------------------------------------------------------
----
>>>
```

Figure 9. Completion examples (13 M).

model was configure to have a more aggressive learning curve. Since both models performed poorly, the problem most likely lies in the dataset. The dataset used was very small and of poor quality. The poor quality stems from the fact that it is comprised of small entries with little overlap and very concise and technical information.



```
○ (.venv)  5:22μμ nick@ArchBTW:~/dev/GitHub/linux_docs_llm % python3 interacti
ve.py checkpoints/20250703_172259-25M/checkpoint_epoch17.pt
Loaded checkpoint: checkpoints/20250703_172259-25M/checkpoint_epoch17.pt
Enter a prompt (or blank to quit).
>>> Arch Linux is an open source
details functionality functionality functionality functionality functionalit
y functionality functionality functionality takes takes take take several se
veral two two two two : : : : : : : : : : : : : : : : : : : : : : : : : : :
: : : : : : : : : : : : : : : : : :
--------------------------------------------------------------------------------
----
>>> To update all packages in Arch Linux use sudo
images pages pages information information information information informati
on information information information information information information s
tructure structure structure functions functions functions functions functio
ns functions functions functions functions functions functions and and and a
nd and and and and and and and and and and and and and and and and set s
et set set set set in in in in in in new new
--------------------------------------------------------------------------------
----
>>>
```

Figure 10. Completion examples (25 M).

6. Conclusion

I demonstrated a theoretically novel approach to generating documentation, which fails in practice. The main factor of this failure is the lack of a comprehensive, clean, and large training dataset. A better approach would be not to pre-train the model from the beginning, but to fine tune it, using the gathered dataset.

References

[1] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *NAACL*, 2019. 1

[2] Jonas Geiping, Micah Goldblum, Tom Goldstein, and et al. Cramming: Training a language model on a single gpu in one day. *arXiv preprint arXiv:2212.14034*, 2022. 1

[3] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NIPS*, 2017. 1